华为天筹求解器技术白皮书（2026年6月）

文档版本 v1.0

华为技术有限公司

编委会

主　编

毛坤

副主编

夏青　郑迥之　朱方舟　王维洲

编写组

杨沐明　安志武　陆嵩　李建树　寸文璟　熊国桢　詹红鑫　王紫菁
赫荣华　苑晗　戴伟民　康钊　林铭倩　戴新晨　张维洋　于程洋
赵成　曾梓涛　代轩　黄泯胜　林轩　童夏良　钟韬　张志宇
陈志堂　袁明轩

前言

求解器作为现代工业软件的核心引擎，是连接数学模型与工程应用的桥梁，广泛应用于运筹优化、控制决策、科学计算等领域，在智能制造、金融风控、能源调度等国民经济关键行业与国防安全领域中具有不可替代的战略地位。求解器是一种基于数学算法的软件系统或算法程序包，旨在通过运筹优化、数值计算等技术，在给定约束条件下寻找复杂工程与科学问题的最优解或精确解。高性能求解器研发是保障产业链供应链安全与韧性的必然要求，同时也是驱动企业智能决策系统升级、实现复杂系统优化与运行效率跃升的技术路径。

当前求解器技术体系涵盖基础理论、智能增强与垂直应用三大维度，逐步从通用算法扩展至行业垂域专业领域。求解器作为工业软件与智能决策的核心引擎，通过将复杂业务问题建模为可计算的优化与数值模型，依托高性能算法驱动关键场景自动决策，提升资源配置效率。然而在迈向规模化应用的进程中面临诸多挑战：将业务问题转化为数学模型的门槛高、NP-Hard问题的计算性能瓶颈、算法稳定性不足、系统集成复杂、成本高昂以及求解策略灵活性有限等问题，制约了求解器在实践中的广泛落地。

面对上述挑战，以华为天筹求解器为代表的先进技术路径，正持续融合超参数优化、策略嵌入与AI辅助算法发现等方法，在数学规划、大规模稀疏矩阵求解等经典与前沿问题上实现国际领先的求解能力，并已在电力调度、智能制造、智慧物流、散热设计等行业实现规模化商业落地。展望未来，随着业务模型复杂度与实时性要求的持续攀升，求解器技术将呈现五大关键演进方向：超大规模问题的求解效率突破、实时决策响应速度的不断压缩、与企业业务流程的深度耦合、AI与传统优化算法的进一步协同增强，以及云化、平台化部署模式的广泛普及，为产业智能化升级注入持续动力。

求解器技术正迎来历史性拐点，华为携带系统性思考以身入局，发布《天筹求解器技术白皮书》，系统梳理求解器的技术框架、核心方法与典型应用，内容涵盖通用求解器与领域求解器的关键技术、华为天筹求解器的实现方案与实践案例，并

对未来技术发展方向进行展望，为产业注入新动能。本白皮书面向多元读者群体，学术界研究人员、产业界伙伴及行业决策者。对于学术界研究人员，书中完整呈现了技术演进脉络、关键原理与前沿趋势，以期激发学术探索与创新；对于科技企业及解决方案供应商等产业界伙伴，书中公开了通用求解器的关键技术，并深入剖析了一批生产调度、批量计划、运输规划、流场仿真等领域求解器的具体技术方案，旨在推动行业技术交流、良性竞争与生态协作，共同提升产业技术水准；对于优化决策相关的行业客户与企业管理者，书中通过典型应用案例展示了求解器技术如何助力解决实际业务中的复杂优化问题并实现业务价值，从而帮助读者理解求解器的技术潜力、评估适用场景、制定智能化业务决策方案。

以求解器为擎，擎动智能未来。本白皮书致力于构筑技术研发与产业应用的坚实桥梁，汇聚产学研智慧，携手全球伙伴共推求解器技术标准化与产品化进程，共同定义下一代求解器的技术范式，让每一次矩阵运算都承载产业变革的重量，让每一个算法都成为自主创新的注脚，共启智能决策新纪元。

目录

概述

求解器技术体系构建了完整的求解能力矩阵，分为通用求解器与领域求解器两大体系。通用数学优化求解器包含数学优化、约束规划和黑箱优化三大方向，分别支持线性/非线性规划、组合优化和无梯度优化等场景。通用数值计算求解器涵盖线性方程组、非线性方程组和特征值求解三大方向，为CAE仿真提供关键算力支持，适用于结构力学、流体仿真等工程领域。随着实际需求的深化以及落地实践的积累，求解器技术与应用正呈现出以下几大关键特征：

1) 复杂度呈指数级增长：实际业务中的模型变量、数据规模急剧增加，已远超单机或传统求解技术的处理上限。客户亟需能够应对海量数据、求解超大规模问题的能力，这要求求解器算法必须持续优化。

2) 实时决策需求日益迫切：市场竞争加剧与环境动态变化推动决策响应时间从数小时乃至数天缩短至分钟甚至秒级，求解效率因而成为核心竞争要素。

3) 求解深度嵌入业务流程：求解器需要无缝集成至企业现有的生产业务系统，成为智能决策的核心组件。

4) AI与求解技术深度融合：通过机器学习预测部分解、自动生成启发式规则，或借助深度学习加速求解初始化、引导搜索策略，结合大模型与进化计算持续增强求解算法。

5) 主导平台化与云化部署：求解器正从独立工具向云服务转型，朝着云原生、SaaS化方向发展，以支持弹性伸缩、降低本地高性能硬件投入、助力用户快速获取最新求解能力。

数学规划求解器是处理线性、非线性及混合整数优化问题的核心工具，其技术体系涵盖算法、架构与工程优化。它通过分层设计实现模型构建、高效求解、结果分析及高阶功能，并能处理多目标、复杂约束与大规模分布式问题。在实际应用中，它为生产调度、物流规划、能源网络等工业优化，资产配置与风险控制等金融问题。当前技术挑战集中在数值稳定性、高维扩展性及通用性平衡，未来将向AI增强、分布式计算与场景定制化方向演进。

约束规划求解器专精于组合优化与离散决策问题，通过建模变量、约束与目标，结合系统化搜索与推理机制来寻找最优或可行解。其主流技术融合了SAT求解、线性规划与启发式策略（CP-SAT框架），通常采用包含应用接口、算法引擎、工具库和系统管理的分层架构，核心算法涵盖预处理、SAT迭代、懒惰子句生成及并行优化等。约束规划在排程、资源分配、路径规划等场景优势显著，尤其擅长处理逻辑复杂、非线性的高维组合问题，能够满足从精确解到快速近似解的全场景需求，是工业智能决策的重要工具。

黑箱优化求解器适用于目标函数未知、无梯度信息的复杂优化场景，仅通过函数评估驱动搜索，可处理连续、离散及混合参数问题。其优势在于不依赖数学模型，通过自适应策略在非光滑、含噪声的高维空间中高效寻优，广泛应用于工业设计、AI调参等领域。典型求解流程包括初始化采样、迭代搜索和收敛输出。关键技术体系主要由三类构成：优化策略、场景适应技术以及工程优化手段。通过算法与工程创新，它为复杂黑箱问题提供了高效、灵活的解决方案，并正向智能化、规模化方向发展。

线性方程组求解器主要分为直接法与迭代法两大类，分别针对中小规模精确求解与超大规模稀疏系统的高效计算。直接法通过矩阵分解获得精确解，适用于千万自由度以下的高精度场景；迭代法通过逐步逼近和预处理技术，能以低内存占用求解自由度达数十亿的稀疏问题。两者常结合为混合策略，在工程仿真、CAE、CFD等领域形成互补，共同支撑复杂工程的高性能计算需求。

非线性方程组求解器是处理结构大变形、湍流耦合等强非线性问题的核心工具，其架构通常基于牛顿法框架，并分为接口层与求解层。接口层适配共享内存与分布式计算，以支持跨节点并行；求解层则通过方向计算和步长控制的协同优化，实现稳定收敛。该求解器依赖底层线性求解器，广泛适用于结构力学、流体力学等领域的复杂仿真。

特征值问题求解器专注于处理大型稀疏矩阵的广义特征值问题，是模态分析、电磁场仿真等CAE应用的关键。它采用分层模块化设计，核心层提供Krylov子空间等方法，以求解极值、区间及高阶特征值；底层则包含谱变换、向量运算等支持模块。该设计兼顾了高效性、稳定性与灵活性，为工程领域提供了一站式特征值分析解决方案。

求解器超参优化技术聚焦于自动调优求解器参数，以针对特定问题分布最小化求解时间、Gap等核心指标。它主要应对高黑盒性、高维依赖与高评估成本三大挑战，其技术路径包括通过代理模型平衡探索与利用的贝叶斯优化、融合大语言模型语义理解能力的智能优化，以及支持多算法并行搜索与可视化监控的推荐框架。该技术的价值在于显著提升求解性能、快速适应冷启动场景，并通过传统优化与AI的融合，为复杂问题提供高效、自动化的参数配置方案。

求解器策略嵌入技术旨在通过深度学习与强化学习方法，直接优化求解器内部的决策模块，以替代传统启发式规则。其核心在于将AI模型嵌入求解流程，例如利用模仿学习优化分支变量选择，采用分层序列模型处理割平面选择，甚至将不同模块的协同建模为博弈问题进行联合学习。这项技术通过机器学习与求解器的深度耦合，实现了决策的智能化与自适应，从而在保证求解质量的同时，大幅提升了计算效率与鲁棒性。

求解器算法发现技术代表了更为前沿的方向，其核心是利用大语言模型（LLM）作为“算法设计师”，在代码逻辑层面自动生成并进化求解策略，如新的启发式规则或搜索逻辑。它主要包含两种实现模式：一是通过标准回调接口与求解器轻量耦合，快速迭代适配特定场景；二是直接深入源码进行深度演化，探索颠覆性的算法重构。这项技术正推动算法设计从“人工经验驱动”迈向“AI自主演化”，为突破求解器性能瓶颈开辟了全新的自动化路径。

与通用求解器面向广泛的基础计算和优化问题不同，领域求解器针对批量计划、生产调度等具体场景，将业务问题转化为数学模型，深度融合行业知识形成智能决策能力。AI技术推动其从单点工具向智能决策中枢转型，通过超参数优化等关键技术实现自学习能力，成为企业智能决策的核心基础设施。

批量计划求解器是应对复杂生产计划的核心智能工具。它通过数学建模与优化算法，在多级产品结构、资源能力与动态需求等复杂约束下，协同优化产能、库存、换型成本与交付目标，旨在实现总成本最低与运营效率最高，推动生产计划从经验判断向数据智能决策转型。

生产调度求解器专注于工序与资源的精确匹配与排序。它处理如作业车间调度等NP-hard难题，在满足工序顺序、机器独占等严格限制下，优化完工时间、设备利用率等多重目标，通过高效的精确或启发式算法，为生产执行提供可操作的详

细调度方案，提升设备利用与交付准时性。

运输计划求解器核心是解决车辆路径规划问题。它旨在对车辆、人员等运力进行科学调度与路径优化，在满足时间窗、载重等业务约束下，寻找成本、时效与装载率的最佳平衡。通过自适应大邻域搜索等智能算法，为城市配送、长途物流等场景生成高效、经济的配送方案，助力物流降本增效。

二维切割求解器聚焦于制造业的原材料节约，核心目标是将一系列不规则零件无重叠地排放在板材上，最大化材料利用率。面对这一几何复杂度极高的NP难题，它依靠计算几何算法处理形状，并采用启发式策略进行优化排样，为钣金、家具、服装等行业提供智能排版方案，显著减少原材料浪费。

三维装箱求解器则专注于解决仓储、运输中的空间优化难题，追求在集装箱或货舱内实现空间利用率最高或容器使用数量最少。它在满足重量、重心、放置规则等复杂约束下，通过多层级贪婪搜索等策略，智能生成最优装载方案，广泛用于物流装载、集装箱运输等场景，有效提升空间利用率和运营经济性。

华为天筹求解器融合数学规划、人工智能与高性能计算，为工业、能源、物流、制造等行业提供智能决策支持，覆盖生产调度、能源管理、物流优化等核心环节，助力企业提效降本。其在航空航线规划、钢铁排产、供应链优化等场景的成功实践，体现了多目标协同优化、高效求解与行业适配性强等优势，为企业提供端到端的智能优化能力，支撑高效、低碳、精细化运营。

以华为天筹求解器为代表的AI求解器，正从传统数学工具向“智能决策中枢”演进。未来，其发展将围绕以下方向展开：

1) 数学优化求解：通过大模型与行业知识库降低自然语言建模的应用门槛；借助GNN、强化学习等AI技术实现算法跃迁，提升求解效率；融合数学规划与约束规划，强化离散优化能力；通过异构计算与云原生架构突破算力瓶颈，实现弹性伸缩与实时决策。

2) 数值计算求解：聚焦算法优化，实现稀疏矩阵并行化、非线性方程智能求解与高精度特征值计算；结合自适应网格与机器学习实现动态路径优化；通过软硬件生态适配构建性价比优势，并以平台化、集成化形态支持跨学科融合。

3) AI辅助求解：依托大模型推动算法从参数调优向自动生成演进；通过实例级策略、白盒代码生成等方式重塑优化路径；重点突破多模态算法设计、工业级适

配与工程级代码生成，最终形成“AI提议+求解器验证”的协同范式，并基于混合架构重构求解器内核。

求解器未来的战略重点在于深化人工智能与数学、物理、运筹学等基础技术的融合，持续推动算法性能突破，构建具有自主知识产权的根技术竞争力。通过有效支撑智能制造、能源优化等关键产业场景，逐步夯实“决策算力”基础设施底座，为工业智能化转型提供核心驱动力。华为将持续深耕天筹求解器的技术研发，依托算法、算力与人工智能的协同创新，积极联合产学研各方力量，将先进的求解能力转化为便捷易用的智能决策工具，助力千行万业有效应对数字化转型过程中的核心挑战，赋能产业实现智能化升级与高质量发展。

本白皮书由华为天筹求解器团队组织编写。毛坤、王维洲负责整体策划、内容设计、大纲制定、审稿定稿、出版发布，撰写前言和概述。郑迥之、夏青、朱方舟分别协调和修订运筹优化、数值计算、AI辅助求解方面内容。第1章求解器技术发展背景由李建树、夏青撰写。第2章通用求解器关键技术中，2.1节数学优化求解技术由杨沐明、郑迥之、寸文璟、陈志堂撰写，2.2节数值计算求解技术由王紫菁、林轩、代轩、曾梓涛撰写，2.3节AI辅助求解技术由朱方舟、钟韬、袁明轩撰写。第3章领域求解器关键技术中，3.1节批量计划求解技术由熊国桢撰写，3.2节生产调度求解技术由苑晗撰写，3.3节运输计划求解技术由戴新晨撰写，3.4节二维切割求解技术由安志武撰写，3.5节三维装箱求解技术由林铭倩撰写。第4章华为天筹求解器技术方案与应用实践中，4.1节航网规划案例由苑晗撰写，4.2节轮胎排产和4.9节医药物流配送案例由詹红鑫撰写，4.3节医药排产案例由康钊撰写，4.4节钢铁板坯计划和4.5节零售门店采购案例由赫荣华撰写，4.6节铝加工排产案例由于程洋撰写，4.7节烟草卷包制丝排产案例由戴伟民撰写，4.8节钣金加工排产案例由熊国桢撰写，4.10节服装排唛案例由安志武撰写，4.11节钢铁配煤和4.12节钢铁能耗案例由李建树撰写，4.13节焊接臂控制案例由张维洋撰写，4.14节铜合金材料设计案例由寸文璟撰写，4.15节电力调度案例由赵成撰写，4.16节云资源调度案例由朱方舟撰写，4.17节汽车外流场仿真案例由陆嵩、黄泯胜撰写，案例分类和排布由张志宇完成。第5章求解器技术未来发展方向和展望，由杨沐明、陆嵩、童夏良撰写。各小节作者同步补充了相应的参考文献、缩略语表和术语定义表。

1 求解器技术发展背景

1.1 求解器定义

在人工智能与科学计算深度融合的背景下，求解器（Solver）被定义为一类专门用于处理大规模数学模型并寻找其数值解的专业软件，承担着复杂问题建模与求解的关键角色[1]。从数学本质上看，求解器是算法逻辑与底层硬件算力之间的桥梁，其任务是将抽象的数学命题转化为计算机可执行的指令流，以获取满足特定精度的数值结果[2]。随着工业、能源、制造、交通等领域的数学计算需求日益增长，传统单一算法已难以满足大规模、高维度、多约束场景下的求解效率与精度要求。例如，在智能工业场景中，生产调度需同时处理时间窗、资源冲突、产能限制等多维度约束；在物流运输中，车辆路径规划需平衡成本、距离与时效性目标；在工程仿真中，非线性方程组与特征值问题的求解直接影响设计结果的可靠性[3]。这些场景的共性在于：问题本质高度非线性或具有组合复杂性，且求解精度和优度与业务收益直接相关，需通过数学建模与算法设计尽量实现最优决策。

为应对这一挑战，现代求解器系统架构逐步演进为通用求解器与领域求解器协同工作的体系。通用求解器聚焦数学规划与数值计算的核心算法，通过标准化接口和高效的计算引擎，为多领域问题提供底层支持；领域求解器则基于通用能力进行场景化适配，解决特定行业中的复杂约束与优化目标。整体求解器系统架构如图1.1所示。同时，AI技术（如判别式机器学习、生成式大模型以及强化学习等）的引入为求解器注入了智能参数调优、策略嵌入与算法发现的能力，显著提升了其在非确定性问题中的适应性与求解效率[4]。这一技术演进推动了求解器从“单点工具”向“智能决策中枢”的转型，成为科学计算与工业优化的核心基础设施。

通用求解器是指能够对数学等式组、不等式组或目标函数进行建模与求解的软件系统，其本质是通过算法实现对复杂问题的数值解析或优化决策。根据技术实现路径与应用目标的差异，求解器可划分为数学优化求解器与数值计算求解器两类。

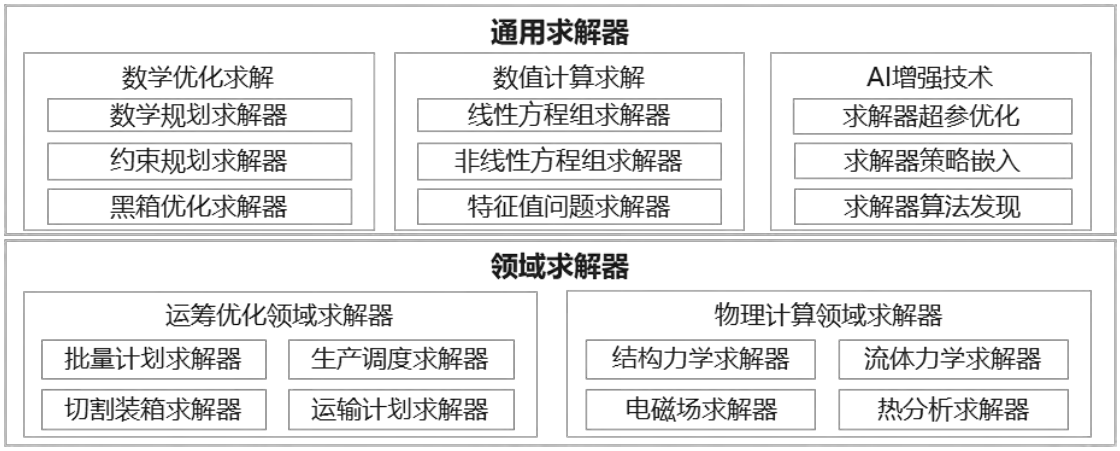

图1.1 求解器架构图

数学优化求解器（Mathematical Optimization Solvers）侧重于决策支持，其目标是在满足一系列复杂约束（Constraints）的前提下，寻找目标函数的最优解（极小值或极大值）。优化求解器的变量空间不仅包括连续实数，还涉及整数、布尔值等离散变量，其核心分类包括：线性规划（LP）、混合整数线性规划（MILP）、非线性规划（NLP）、混合整数非线性规划（MINLP）、约束规划（CP）等[5]。数学优化求解器旨在通过数学规划方法（如线性规划、整数规划）或启发式策略（如遗传算法、领域搜索）寻找最优解，适用于资源分配、生产调度、路径规划等离散或混合问题。其技术特征包括规划算法的多样性，涵盖了线性规划（LP）、二次约束二次规划（QCQP）、混合整数规划（MILP/MIQP）和约束规划（CP）。在线性规划方面，数学优化求解器可以支持单纯形法与内点法，并兼容无界、不可行问题的检测与灵敏度分析；二次约束二次规划针对凸问题提供精确解，非凸问题则支持启发式近似；混合整数规划通过分支定界和割平面算法处理离散变量，并支持不可约不可行子系统（IIS）分析与松弛问题求解；约束规划则以布尔变量和整数变量为基础，结合全局约束（如AllDifferent、Cumulative）实现高效的组合问题搜索。数学优化求解器的主要应用领域包括但不限于工业制造中的生产计划与排程、切割与下料优化、资产组合优化、能源网络设计、运输路径规划、资源调度、供应量与库存管理等前沿领域。

数值计算求解器（Numerical Computing Solvers）主要处理连续数学中的基础代数问题，其核心目标是解决形如 $Ax = b$ 的线性方程组、求解特征值问题 $Ax = \lambda Bx$ 以及处理超定系统的最小二乘拟合问题。这些问题构成了现代工程模拟的物

理底座，是微分方程离散化后的最终计算形态。数值计算求解器的核心功能包括求解线性与非线性方程组、特征值问题等数值分析任务，广泛应用于工程仿真、物理建模与连续系统优化场景[6]。其技术特征表现为：一是算法覆盖性广，支持线性问题求解的直接法（如LU分解、Cholesky分解等）、迭代法（如共轭梯度法、Generalized Minimal Residual Method，GMRES等）和特征值分解，以及非线性问题求解的迭代法（牛顿法、拟牛顿法等）；二是稀疏矩阵的高效处理，针对大规模稀疏问题（如有限元仿真离散问题），采用CSR、CSC等高效存储格式，结合重排序、填充控制等技术降低计算复杂度[7]；三是灵活的收敛性控制，通过动态调整残差阈值、迭代步长、预处理策略（如ILU、AMG）实现数值稳定性与收敛效率的平衡；四是多场景适配,支持标准特征值问题、广义特征值问题，以及多阶特征值计算（如Krylov-Schur方法、FEAST算法）；五是并行与分布式计算，兼容单核、多线程（如OpenMP）、多节点分布式（如MPI）等计算模式，适应从个人计算机到高性能计算集群的部署需求[7]。数值计算求解器的主要应用领域包括但不限于计算机辅助工程仿真——例如结构力学分析、流体动力学模拟、热传导计算和多物理场耦合问题等。它同样广泛应用于计算物理学（如等离子体模拟、量子力学计算）、地球科学（如地震波传播、地下水流动建模）、计算化学（如电子结构计算）、生物医学工程（如心脏电生理仿真）等前沿领域[8][9][10]。

AI辅助求解技术的核心在于利用人工智能方法增强传统求解器在建模、搜索与超参优化过程中的智能性与自适应能力，其重点并不在于替代经典数值或组合算法，而是通过数据驱动和学习机制对求解流程进行感知、预测与引导，从而提升求解效率、鲁棒性与泛化能力[11]。该技术主要可划分为三大核心方向：一是超参优化，即基于历史求解经验或问题特征，利用元学习、贝叶斯优化或强化学习自动配置求解器内部参数（如分支策略、启发式权重、时间分配等），以适配不同问题实例；二是策略嵌入，指将神经网络、图神经网络或注意力机制深度集成到求解器的关键决策模块中，例如用学习模型替代手工设计的变量选择规则、割平面生成策略或冲突分析逻辑，使搜索过程更具上下文感知能力；三是算法发现，即通过程序合成、符号回归或大语言模型驱动的自动推理，探索全新的求解范式、启发式规则甚至完整算法结构，突破人类专家设计的局限[12]。AI辅助求解技术的典型特征包括对问题结构的隐式学习能力、对求解状态的动态响应机制、对多源异构数据（如问

题实例特征、求解轨迹和失败日志）的融合利用，以及在保持数学严谨性前提下的灵活可扩展架构。目前，该技术已在大规模组合优化（如物流调度、芯片布线）、科学计算（如微分方程求解、材料模拟）、运筹决策（如能源管理、金融风控）以及形式化验证（如程序合成、智能合约检测）等领域展现出显著潜力，正逐步成为下一代高性能求解器不可或缺的关键特性[13]。

领域定制化求解器技术的核心在于针对特定行业或应用场景的结构化问题特征，对通用求解框架进行深度适配与优化，其侧重并非追求算法的普适性，而是通过融合领域知识、问题语义与业务约束，构建高效率、高精度和高可靠性的专用求解能力，从而在复杂现实约束下实现可落地的最优或近优决策[14]。这类求解器通常围绕典型工业优化场景形成若干核心分类：面向流程制造与离散制造的批量计划求解器，聚焦多阶段、多资源、多约束下的生产批量与时间窗协同优化；生产调度求解器，强调工序排序、设备分配、换模时间与柔性作业路径的实时动态调度；运输计划求解器，处理多车场、多车型、时间窗、载重与路线连通性等组合约束下的车辆路径与配送优化；二维切割求解器，针对板材、玻璃、布料等材料的矩形或异形件排样问题，最大化材料利用率并满足工艺限制；以及三维装箱求解器，解决物流、仓储和集装箱装载中多尺寸物品的空间堆叠、重心平衡与装入顺序优化难题。这些定制化求解器的技术特征体现为对特定问题结构（如对称性、稀疏性、分层依赖）的显式建模、对行业规则（如安全间距、工艺顺序、装载稳定性）的硬编码或软约束集成、对大规模实例的高效启发式与精确算法混合策略，以及与企业信息系统（如ERP、MES和WMS）的深度耦合能力。目前，领域定制化求解器已广泛应用于智能制造、供应链管理、物流运输、航空航天、家具建材和快销品生产等对资源利用效率与运营成本高度敏感的行业，成为支撑数字化转型与智能决策的关键基础设施。

1.2 求解器价值

求解器的核心价值在于其将数学算法转化为可复用、可扩展的工业能力。高性能求解器通过集成复杂的数学理论（如对偶理论、割平面技术），能够在解的质量（最优性界限）与收敛速度上提供严谨的保证。对于复杂的决策优化问题，专业求

解器利用深度的底层优化和并行计算技术，能将决策耗时从数天降低至数分钟，这种效率的跨越式提升是现代供应链与智能制造得以运行的前提。求解器的价值已超越传统数学工具范畴，演变为支撑现代工业体系的核心使能技术[15]。

1) 求解技术的战略价值

在科学计算与工业智能化深度融合的背景下，求解器已演变为支撑现代工业体系的核心技术基础设施。从底层计算支撑到高层决策优化，其价值呈现多维度特征。

数学优化求解器已成为资源约束下的决策中枢。电力调度系统需在分钟级完成全国机组组合的实时调配，港口智能计划通过混合整数规划（MILP）模型协调船舶、龙门吊与堆场作业，提升吞吐能力。专业求解器相较于传统依赖人工经验或规则决策方式，在解的质量（最优性界限）与求解效率上具有显著优势。

数值计算求解器构成了复杂物理问题解析的基石。在工程仿真中，有限元分析的刚度矩阵求解、流体动力学的控制方程离散化，均依赖高性能线性求解器。通过稀疏矩阵存储（CSR/CSC）与预处理技术（如ILU、AMG），可将亿级自由度问题的求解时间从数小时压缩至分钟级，显著提升研发效率。这种计算能力的突破，使得高频电路设计、非线性材料分析等复杂工程问题得以实现高精度解析。

2) 自研核心技术的战略需求

当前国际竞争格局下，自研求解器具有双重战略意义：一方面突破“卡脖子”技术封锁，另一方面构建自主可控的工业软件生态。技术自主性要求覆盖全技术栈，接口层需支持C++/Python等多语言SDK及MPS/LP标准格式；算法层需突破病态矩阵处理、不可行问题诊断等关键技术；架构层需实现MPI与OpenMP混合并行，适配千节点级分布式计算。

3) 客户应用的深度价值创造

求解器的价值最终体现在行业场景的深度赋能。在能源领域，运输计划求解器通过构建车辆油耗与载重的动态约束模型，生成低碳路径方案，可以大幅度降低物流碳排放。金融行业应用混合整数二次规划（MIQP）求解器，实现投资组合的收益-风险平衡优化。制造业典型应用显示，柔性作业车间调度（FJSP）通过混合整数规划可以大幅度缩短完工时间；动态排产系统结合约束规划（CP）与遗传算法，可以降低设备空闲率。这些实践表明，专业求解器通过算法与业务特性的深度适配，

正在重塑制造业的决策范式[16]。

4) 技术演进与生态构建

求解器的未来发展呈现三大趋势：首先是AI融合增强，如Xpress求解利用AI策略实现了放缩策略的自动选择；其次是领域求解器专业化，形成"通用内核+场景适配"的生态体系；此外还有分布式计算扩展，支撑5G基站布局优化等亿级变量问题求解。这些技术演进正在推动工业软件从工具向基础设施升级。通过开放API接口，接入工业平台，形成覆盖研发设计、生产运营的全链条优化能力。这种生态化发展路径，不仅降低了企业使用门槛，更构建起可持续的技术演进体系。

1.3 求解器技术架构

求解器技术架构是支撑其高效性、通用性与智能化的核心框架，其设计融合了数学优化、数值计算与人工智能三大类技术体系[11][16][17]，因此整体架构可分为数学优化技术体系、数值计算技术体系与AI增强技术体系三大模块，各模块通过标准化接口实现数据交互与功能协同。以下分别展开说明。

1.3.1 数学优化技术体系

数学优化技术聚焦决策优化问题，包含三大类求解器：

1) 数学规划求解器

数学规划求解技术体系以问题结构识别为基础，深度融合了凸分析、非线性规划、组合优化与数值计算方法。该体系首先通过自动检测目标函数与约束的性质（如线性、二次、锥形、整数或一般非线性）对问题进行分类，进而调用相应的专用算法框架：对于线性规划和混合整数线性规划，以内点法与单纯形法为连续松弛核心，结合分支定界、割平面、启发式搜索与冲突分析实现全局最优；对于二次规划和二阶锥规划，则依托内点法处理凸结构；而对于非线性规划和混合整数非线性规划，则采用序列二次规划、增广拉格朗日法、广义Benders分解或外逼近等策略，在局部收敛性与全局探索之间取得平衡。整个体系高度强调鲁棒性、精度与可扩展性，不仅集成先进的预处理、缩放、对偶信息利用和热启动机制以加速求解，还通过并行分支、分布式割生成和自适应参数调优应对大规模复杂优化问题，从而为能

源、金融、制造、物流等领域的智能决策提供可靠、高效的数学支撑。

2) 约束规划求解器

在数学优化体系中，约束规划求解器以离散组合问题的结构化推理为核心，构建了一套融合逻辑传播、搜索控制与领域知识的求解技术体系。该体系不依赖于目标函数的连续性或可微性，而是将问题建模为一组变量及其取值域上的约束网络，通过高效的约束传播机制在搜索过程中动态缩减变量域，提前剪除不可行区域；同时，结合智能搜索策略、冲突驱动的回溯学习、以及懒惰或多点搜索等高级技术，显著提升求解效率。现代约束规划求解器还深度集成可满足性、混合整数规划和局部搜索等方法，形成混合求解范式，并支持全局约束的符号化处理以捕捉问题高层结构，从而在调度、排班、配置、资源分配等强离散、高组合复杂度的场景中实现精确而高效的求解能力。

3) 黑箱优化求解器

在数学优化体系中，黑箱优化求解器面向目标函数和约束无法提供解析表达式或梯度信息的复杂场景，构建了一套以采样、代理建模与智能探索为核心的无梯度优化技术体系。该体系不依赖问题的内部结构，仅通过有限次的输入-输出评估来驱动优化过程，广泛采用响应面方法、高斯过程、径向基函数或深度神经网络等代理模型对未知函数进行局部或全局近似，并结合采集函数平衡探索与利用；同时融合直接搜索策略、进化算法、粒子群优化及随机局部搜索等元启发式方法，以应对非凸、非连续、多峰甚至含噪声的黑箱目标。整个体系强调样本效率、鲁棒性与自适应能力，支持并行评估、约束处理、多目标权衡及热启动机制，能够在工程设计、超参调优、实验自动化等“仿真即计算”或“实验即评估”的高成本黑箱场景中，以尽可能少的函数调用实现高质量解的高效发现。

1.3.2 数值计算技术体系

该体系聚焦连续物理问题离散的数学问题求解，重点突破计算效率与精度瓶颈：

1) 线性方程组求解器

通过矩阵结构分析（如对称性、正定性、稀疏模式）和谱性质评估，引导求解策略的选择；在此基础上，构建由预处理子、Krylov子空间迭代法或直接法组成的

协同求解框架。其中，预处理技术尤为关键，涵盖代数多重网格、不完全分解、块Jacobi、偏转等方法，旨在改善系统矩阵的条件数与特征值分布，从而加速收敛。整个体系强调可扩展性与鲁棒性的统一，不仅需支持从单机到万核级并行架构的高效执行，还需针对多物理场耦合、奇异或病态系统等复杂场景提供自适应策略，最终实现对大规模、高维、非线性离散化问题背后线性子问题的快速、稳定、可靠求解。

2) 非线性方程组求解器

非线性方程组求解技术体系以牛顿类方法为核心，融合了局部线性化、全局收敛策略与高效线性子问题求解机制。该体系通常从对非线性残量函数的可微性与结构特性分析出发，采用牛顿法或拟牛顿法构建局部超线性或二次收敛框架，并通过线搜索、信赖域或延续法等全局化策略确保迭代过程在远离初值时仍能稳健收敛；其中，每一步迭代所依赖的雅可比矩阵或其近似形式，直接决定了算法的精度与效率，而对应的线性系统则交由高性能线性求解器处理，形成“非线性-线性”两级嵌套求解架构。整个体系高度强调自适应能力与鲁棒性，能够根据残量下降行为、雅可比条件数或物理约束动态调整步长、更新策略或求解精度，并支持大规模并行计算，从而有效应对多物理场强耦合、材料非线性、接触不连续等复杂工程场景中的高维非线性挑战。

3) 特征值问题求解器

特征值求解技术体系围绕矩阵谱结构的高效近似与目标特征对的精准提取而构建。该体系依据问题规模、矩阵性质及求解目标选择合适的算法框架，主要包括子空间迭代法、Lanczos方法、Arnoldi方法以及基于谱变换的加速策略。为提升收敛效率与鲁棒性，现代特征值求解器普遍集成精化技术、动态重启机制以及高性能线性求解器作为内核支撑，以高效处理位移求逆所需的线性系统。整个体系强调可扩展性与自适应性，不仅支持大规模稀疏矩阵在分布式内存环境下的并行计算，还能结合物理先验知识（如多尺度基函数或约束子空间）引导谱搜索方向，从而在结构动力学、量子化学、稳定性分析等关键领域实现对高维特征问题的可靠、高效求解。

1.3.3 AI 增强技术体系

AI增强技术为求解器注入了新的能力。与传统方法相比，AI技术通过数据驱动策略提升了算法的效率和鲁棒性。本节将从三个层次介绍AI增强技术：超参优化系统、策略嵌入技术和算法发现引擎。

1) 超参优化系统

超参优化系统是AI增强技术体系的重要组成部分。它连接算法能力与实际应用效能，融合了贝叶斯优化、元学习、多保真度评估与自动化反馈机制，形成了一套智能调优技术体系。该体系的核心目标是最小化目标模型在验证集上的性能评估代价。首先，通过高斯过程或树结构代理模型对超参-性能映射关系进行概率建模。然后，利用采集函数动态指导高潜力超参的采样。同时，引入多保真度优化技术，在不同资源预算下快速筛除劣质配置，显著提升样本效率。进一步地，系统通过元学习积累历史任务的调优经验，构建跨任务的知识迁移能力，实现冷启动加速；结合强化学习或进化策略处理离散-连续混合搜索空间，并支持多目标权衡。

2) 策略嵌入技术

在AI增强技术体系中，策略嵌入技术旨在将数据驱动的智能决策能力深度融入传统数学求解器的核心算法流程，构建了一套以学习引导、自适应调控与混合推理为特征的融合式求解架构。该技术体系通过离线或在线方式，利用强化学习、图神经网络、元学习或模仿学习等方法，从大量历史求解实例中提取问题结构与求解行为之间的映射规律，将学习到的策略以轻量级模型或规则形式嵌入到求解器的关键决策点。这些决策点包括：预处理子选择、粗网格构造、搜索方向调整、分支变量排序、步长控制或迭代终止判断等环节；所嵌入的策略不仅能感知当前问题的代数或组合特征，还能动态响应求解过程中的状态反馈，实现对传统确定性和启发式规则的智能化替代或增强。整个体系强调可解释性、泛化性与低侵入性，在保持求解器数值稳定性与收敛保证的前提下，显著提升其在复杂、异构或高维问题上的鲁棒性与效率，从而推动求解器从“通用工具”向“场景自适应智能引擎”的演进。

3) 算法发现引擎

在AI增强技术体系中，算法发现引擎代表了一种面向未来求解器自主演化的前沿范式，其技术体系以程序合成、神经符号推理与演化学习为核心，致力于从问

题语义、性能反馈与数学先验中自动设计或重构高效求解算法。该体系不再局限于对现有算法的参数调优或策略嵌入，而是将算法本身视为可学习、可生成的对象，通过结合形式化语言（如DSL）、图神经网络对问题结构进行编码，并利用强化学习、进化搜索或大语言模型驱动的程序生成机制，在算法空间中探索满足收敛性、复杂度与精度约束的新型求解流程；同时，依托大规模求解日志与跨任务元知识库，引擎能够识别通用计算模式（如迭代模板、预处理组合、多级校正结构），并通过符号回归或神经程序归纳提炼出可解释、可复用的算法组件。整个体系强调“发现-验证-泛化”闭环，集成形式验证确保生成算法的数值正确性，并支持人机协同迭代优化，从而为科学计算、组合优化与微分方程求解等领域持续输出创新性、高适应性的下一代求解策略。

1.4 求解器技术现状

求解器作为支撑科学计算、工程仿真与智能决策的核心工具之一，其技术发展呈现出算法革新与跨领域融合的双重特征。当前主流求解器技术可分为数学规划、约束求解、微分方程求解三大方向，同时人工智能技术正在深度重构传统求解范式。

在数学规划求解领域，传统优化问题的求解效率持续突破。以Gurobi、CPLEX为代表的商业求解器通过算法层面的创新实现性能跃升，例如Gurobi 10.0版本通过优化单纯形法与内点法的数值稳定性，并引入启发式和割平面等多项算法创新，使大规模混合整数规划（MIP）问题的求解速度提升超过30%[16]。开源求解器如SCIP和HiGHS则通过改进分支定界策略和并行计算架构，在超大规模线性规划（LP）问题中展现出竞争力。值得关注的是，华为OptVerse求解器在特定应用场景中已实现关键技术突破，其基于自研硬件架构优化的并行计算模块，在供应链优化等场景中展现出独特的性能优势。

科学计算领域的数值计算求解器技术已形成以高性能、可扩展与鲁棒性为核心的成熟体系：PETSc通过模块化设计支持Krylov子空间方法、丰富预处理子（如AMG、Deflation、Block Jacobi）及非线性/特征值求解器的深度组合，强调用户对求解流程的细粒度控制[18]；而Hypre则以高度优化的代数多重网格（BoomerAMG）为核心，在万核级并行环境下展现出卓越的弱/强可扩展性[19]。二者共同推动了从

单机到超算平台的高效求解能力，但在面对复杂多物理场耦合、病态系统或非结构化网格时，仍依赖专家经验进行调参，自动化与自适应能力有限。近年来，以华为OPTVerse-NCSolver（下文简称为NCS）为代表的数值计算求解器实现了PETSc、Hypre和Pardiso等的核心能力，更通过融合AI增强技术显著提升了求解器在未知问题上的泛化性与易用性，标志着我国在底层求解引擎领域正从“可用”迈向“好用”与“智能”的新阶段。

人工智能与求解器的双向赋能正在形成技术闭环。当前，AI赋能求解器技术正从“辅助调参”迈向“内生智能”的深度融合阶段，核心趋势在于将机器学习、图神经网络与强化学习等能力嵌入求解器的决策内核，实现对算法选择、参数配置、预处理构造及迭代策略的自适应优化：例如利用历史求解数据训练模型以预测最优预处理子（如AMG粗化策略或Deflation子空间）、通过图表示学习捕捉矩阵结构特征以指导Krylov方法重启频率、或借助贝叶斯优化自动调谐非线性求解器的步长与容差，从而显著提升求解效率与鲁棒性，尤其在面对多尺度、强耦合或病态系统等传统方法难以泛化的复杂场景中展现出独特优势。在此方向上，求解器技术已取得突破性进展，以华为天筹求解器为代表的全能力AI+求解器引擎，创新性地将人工智能深度融入优化求解和数值求解全流程，构建了“AI+求解”双轮驱动的新技术体系，实现从建模到求解的自动化闭环，不仅在供应链生产计划、航班调度等超大规模工业场景中验证了性能超越国际主流工具的潜力，更标志着我国在AI原生求解器这一前沿赛道上实现了从跟随到并跑乃至局部领跑的战略跃升。

1.5 求解器技术挑战

在求解器技术的发展与应用过程中，尽管算法和硬件不断进步，但仍面临一系列深层次、系统性的技术挑战。这些挑战不仅制约了求解器在复杂现实问题中的性能与可靠性，也构成了当前科研与工程攻关的重点方向。主要包括以下几个方面：

数值稳定性问题：求解器在处理大规模或病态问题时，极易受到浮点运算舍入误差的影响，导致解的精度下降甚至完全失效。例如，在线性规划中，单纯形法对基矩阵的条件数高度敏感；在微分方程求解中，刚性系统若未采用合适的隐式方法，可能引发数值震荡或发散。此外，混合整数规划中松弛子问题的数值误差可能误导

分支决策，造成求解失败或次优解[20]。提升数值鲁棒性需结合高精度算术、预处理缩放、正则化策略及误差传播分析等手段，但往往以牺牲计算效率为代价。

维数灾难问题：随着问题规模（变量数、约束数或状态空间维度）指数级增长，求解器的内存消耗与计算时间急剧上升。在组合优化中，MIP问题的可行解空间呈组合爆炸；在动态规划或PDE求解中，状态/网格点数量随维度呈指数增长。传统精确算法难以扩展至高维场景，迫使研究者转向分解、采样、降维等近似方法，但这些方法可能牺牲最优性或理论保证[21]。如何在可扩展性与解质量之间取得平衡，是高维求解的核心难题。

通用性与专用性间的平衡问题：尽管AI驱动的智能求解策略展现出潜力，但其泛化能力仍受限。训练数据依赖特定问题分布，导致模型在新领域表现不稳定；同时，黑箱决策机制削弱了求解过程的可解释性与可信度。求解器需在“通用求解框架”与“领域定制优化”之间权衡[15]，过度通用导致效率低下，过度定制则丧失迁移能力。如何构建具有自适应学习能力且保持数学严谨性的智能求解架构，仍是开放问题。

非凸性与多模态优化难题：现实问题常涉及非凸目标函数或非线性约束，导致存在大量局部极小值。传统基于梯度或凸松弛的方法易陷入次优解，而分支定界、区间分析等全局优化算法计算成本极高[22]。此外，多目标、不确定或动态环境下的优化进一步加剧了求解复杂性，要求求解器具备更强的探索-利用平衡能力和在线重规划机制。

并行与分布式扩展瓶颈：虽然现代求解器普遍支持多线程或分布式计算，但在实际并行效率上仍面临挑战，例如，MIP的树搜索天然存在负载不均衡与通信开销；SAT求解的冲突分析难以有效并行化；微分方程求解在GPU上的细粒度同步成本高等问题[23]。如何设计可扩展的并行原语、减少冗余计算和实现异构计算资源的高效协同，是提升大规模求解能力的关键。

建模与求解间的交互问题：用户建模方式（如变量定义、约束形式）极大影响求解效率，但多数用户缺乏“可求解性感知”的建模经验，求解器也难以自动识别模型结构弱点（如对称性、弱松弛、冗余约束）并进行有效重构。自动预处理、模型诊断与反馈机制的缺失，使得建模质量和求解性能高度依赖专家人工优化，限制了求解器在工业自动化场景中的落地。

2 通用求解器关键技术

通用求解器是解决非限定应用背景下的通用数学模型问题的工具。通用求解器技术按照技术体系可分为数学优化、数值计算和人工智能三大类。本白皮书将在本章中展开介绍数学优化、数值计算，以及人工智能在求解器中的重点应用——AI辅助求解三方面技术。

2.1 数学优化求解技术

数学优化是解决带有优化目标的数学模型的技术。数学优化技术在运筹学、工程、金融、物流等领域具有重要的应用。通过优化资源分配、生产过程等决策，数学优化可以显著降低开销并提升工作流程的效率[24][25]。本白皮书将重点介绍数学优化中应用最广泛的三类问题：数学规划问题、约束规划问题和黑箱优化问题，以及它们的主流求解器技术。

2.1.1 数学规划求解器技术

本节将从数学规划问题的定义和分类，数学规划求解器软件架构，数学规划求解器功能，以及最常见的数学规划问题类型及其关键求解技术若干方面对数学规划求解器技术进行介绍。

2.1.1.1 数学规划简介

数学规划（Mathematical Programming，MP）是数学优化中的一个分支，它的主要研究目标是在给定的区域中寻找可以最小化或最大化某个或某几个函数的最优解[26]。数学规划问题的可定义为如下的通用形式：

$$\min \ f(x)$$

$$s.t. \ \ g(x) \leq h, x \in X$$

其中 f 称为数学规划问题的目标函数（Objective Function），g 称为约束条件（Constraint），x 称为决策变量（Decision Variable）。数学规划问题被广泛应用

于生产调度、金融投资、能源管理、智能制造等领域，是实现资源高效配置与智能决策的关键工具。考虑目标函数、约束条件和决策变量的不同类型，数学规划问题可进一步划分成许多不同分支，例如：

1) 仅包含线性约束和目标函数的线性规划（Linear Programming，LP）；

2) 二次目标函数或约束的二次约束二次规划（Quadratic Constrained Quadratic Programming，QCQP）；

3) 带有凸二阶锥约束的二阶锥规划（Second Order Cone Programming，SOCP）；

4) 包含一般非线性约束或目标的非线性规划（Nonlinear Programming，NLP）；

5) 考虑整数型变量的混合整数线性规划（Mixed Integer Linear Programming，MILP）和混合整数二次约束二次规划（Mixed Integer Quadratic Programming，MIQP）。

数学规划求解器（Mathematical Programming Solver）就是包含了某类或某几类数学规划问题求解算法的软件包。按照求解目标进行划分，数学规划求解算法可以分为精确算法和启发式算法。精确算法从理论上能够为数学规划问题提供严格的正确性保证：对于给定问题，在算法适用条件满足时，可以判定其为最优、不可行或无界；若问题可行且存在全局最优解，则能够求得至少一个全局最优解。启发式方法有可能找到问题的可行解，但不保证解的质量和最优性。精确方法的计算时间通常较长，适用于中小规模问题。启发式方法通常具有较高的执行效率，适用于大规模且约束条件复杂的问题。常见的精确方法有单纯形法、内点法、分支定界法。常见的启发式方法有贪婪搜索方法、圆整方法、邻域搜索。本白皮书将重点介绍各类数学规划问题的精确算法。除了最核心的求解算法技术之外，本白皮书还将涉及数学规划求解器的架构设计和功能介绍。

2.1.1.2 数学规划求解器软件架构

如图2.1所示，数学规划求解器架构可分为四个部分：交互接口、核心算法引擎、通用模块和底层能力。交互接口层作为用户与系统交互的桥梁，提供多维度的求解器调用方式：

1) 多语言软件开发套件（Software Development Kit，SDK）支持：通过C++、Python、Java等主流编程语言接口，开发者可快速集成求解器功能；

2) 云服务接口：支持分布式计算环境下的远程调用与资源调度；

3) 交互式命令行（Shell）：提供即时命令执行与调试功能；

4) 命令行工具：满足自动化脚本与批处理需求。

该设计实现了从开发环境到生产环境的全场景覆盖。引擎核心层作为系统的核心计算中枢，针对五类主流数学规划问题提供相应的求解算法支持：

1) 线性规划：采用单纯形法或内点法求解；

2) 混合整数线性规划：基于分支定界算法实现离散变量隐枚举求解；

3) 混合整数二次规划和混合整数二次约束规划（Mixed Integer Quadratic Constrained Programming，MIQCP）：结合二次（约束）规划与分支定界的求解算法；

4) 二次约束二次规划：处理二次目标函数或约束的内点法；

5) 二阶锥规划：支持锥约束的内点方法求解；

6) 一般非线性规划：求解一般非线性模型的局部最优算法。

通用模块层提供系统级支撑功能：

1) 参数管理系统：统一配置管理与调参；

2) 日志管理：求解迭代过程日志展示和最终结果输出；

3) 文件IO：支持MPS等多种格式的模型输入输出；

4) 事件管理：并行同步等事件处理与状态监控；

5) 异常处理机制：多级错误捕获与恢复策略；

6) 回调接口：用户自定义算法和控制逻辑注入点。

这些模块通过标准化接口实现与核心算法的解耦。底层能力层是构建高性能计算基础：

1) 矩阵计算引擎：优化线性代数运算性能；

2) 并行计算框架：支持多核/分布式计算资源调度；

3) 基础算法库：包含排序、搜索等核心数据结构操作。

该层通过底层优化实现算法效率和稳定性的显著提升。

图2.1 数学规划求解器软件架构

2.1.1.3 数学规划求解器求解流程

如图2.2所示，数学规划求解器的整体求解流程可概括为：模型初始化、预处理、算法选择、算法执行、后处理、返回结果。其核心逻辑是通过系统化的步骤，将实际问题转化为可计算的数学模型，再通过高效的算法逐步逼近最优解，最终为决策提供支持。不同类型的数学规划问题（如线性规划、整数规划）会在算法选择和迭代逻辑上有所不同，但整体框架保持一致。

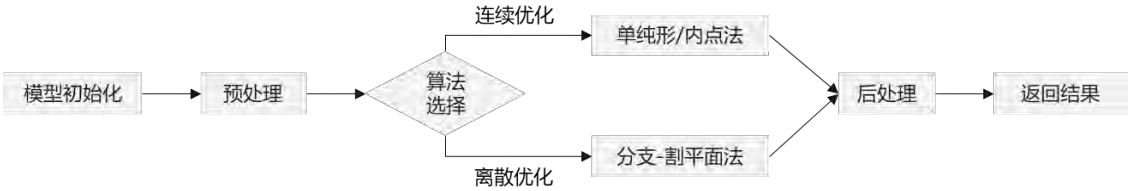

图2.2 数学规划求解器求解流程

2.1.1.4 数学规划求解器功能特性

数学规划求解器作为用户搜索数学规划问题最优解的工具，应提供基础的数学建模、参数设定、求解、结果获取等功能。同时，也应提供回调、模型不可行性分析等高阶功能。主流数学规划求解器应支持如下所述功能：

1) 模型支持：应支持主流数学规划的建模和求解，支持一般整数变量、二元变量、连续变量及变量类型的任意组合、支持等式或不等式线性约束、支持特殊有序

集约束、指标约束、最大最小值约束、绝对值约束等特殊约束，支持上述约束类型的任意组合，支持常量或线性目标函数，支持多个目标函数，支持极大化或极小化问题，支持可行域有限、无限（无界）或为空（不可行）的问题；

2) 求解能力：精确求解器，内置单纯形法、内点法等线性或非线性连续规划问题求解，支持分支定界、割平面、启发式等混合整数线性规划求解器常用功能模块，确保在资源允许范围内给出可证最优解，或证明问题的不可行性或无界性；应支持不可行问题的不可行原因分析功能，包括找出不可约不可行子系统或给出可行性松弛；

3) 数据输入：应支持数学规划常用模型文件（如MPS、LP及其压缩格式）的文件读写功能；

4) 日志和结果输出：应支持求解信息打印，包括输入模型简要信息、预处理后模型简要信息、使用的线程数、节点求解进度（如已求解节点数、待求解节点数、平均LP单纯形迭代次数、当前最优解和最优界、相对最优间隙），求解结果汇总（如总时间、最优解和最优界、相对最优间隙、总节点数、总LP迭代次数）；

5) 参数设置：应支持求解时间上限、内存使用上限、最大可用线程数、相对最优间隙容忍度、可行性容忍度、随机种子等常用参数设置；应支持多种模型预处理策略，割平面策略，启发式策略，并可通过参数进行开关或程度控制；

6) 功能接口：应支持预处理后模型导出、初始解设置、分支变量优先级设置、可行解池、回调函数等功能；应支持中断求解，并返回中断时的结果；

7) 并行加速：应支持多线程以及多节点分布式求解，应支持确定性并行求解。

8) 结果获取：应支持常用结果信息获取，如问题求解状态、最好解目标值、其余可行解、解的违反度、相对最优间隙，支持解文件（SOL）的输出。

2.1.1.5 线性规划关键技术

线性规划是运筹学中用于优化问题的经典方法，广泛应用于资源分配、生产计划、物流调度等领域[27]。它通常被定义为在满足一组线性约束条件下，求解线性目标函数的最大值或最小值，数学标准形式为：

$$c^* = \min\{c^T x \mid Ax \le b, x \in \mathbb{R}^n\}$$

其中，x 为决策变量向量，c 为目标函数系数，A 为约束矩阵，b 为资源限制向

量。线性规划算法是数学规划的基石，从变量属性和目标或约束函数特征角度，可以扩展到混合整数线性规划、二次约束二次规划及一般非线性规划范畴。

如图2.3所示，线性规划问题是基础的问题类型，线性规划问题的求解算法也是各类数学规划问题求解算法的基础。LP核心算法包括单纯形法（Simplex Method）和内点法（Interior-Point Method），其中单纯形法通过高效搜索多面体顶点寻找LP问题最优解，而内点法通过高效搜索多面体内部迭代逼近LP问题最优解。单纯形法通常适合中小规模问题，而内点法具有多项式复杂度，适合大规模稀疏问题。在实际问题求解中，内点法也可能比单纯形慢，这是由于需要处理复杂的矩阵计算。除了算法选择外，预处理技术也是LP求解的关键环节，其核心目标是简化问题结构、减少计算量、提高数值稳定性，并提前检测问题性质（如无解、无界）。

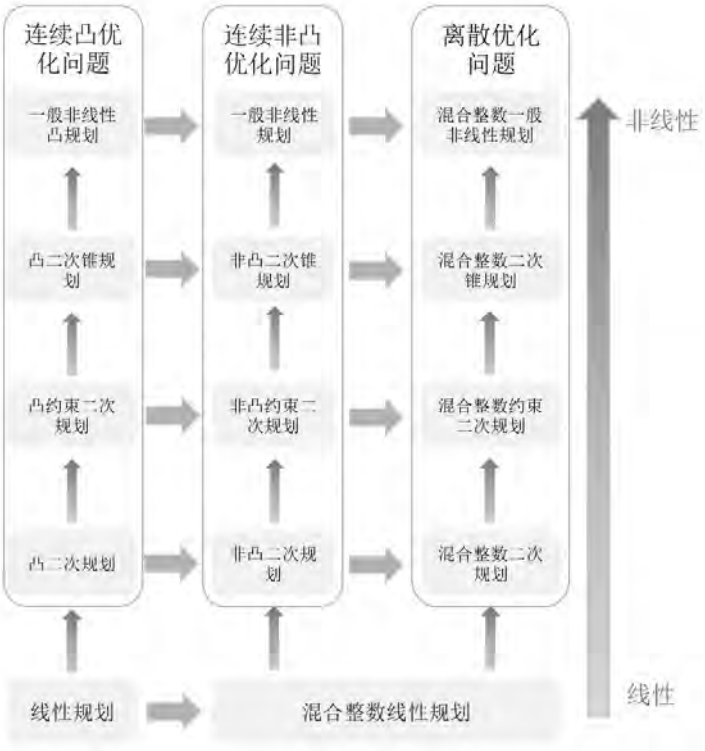

图2.3 数学规划问题分类

（一）单纯形法

单纯形法是一种经典的算法，它的原理是通过在可行域的顶点（即基可行解）上迭代，每次迭代选择一个进入基的变量和一个离开基的变量，逐步逼近最优解[28]。按照对偶理论，单纯形求解器又包含原始和对偶单纯形算法，核心区别在于迭代方向和初始解的可行性：

1) 原始单纯形法在保持原始可行性的基础上优化目标值，在可行解构成的几何多面体顶点上移动，逐步爬升（或下降）目标值；

2) 对偶单纯形法保持对偶可行性，修复原始不可行性，在“对偶可行”的空间中移动，逐步将不可行的原始解拉回可行域。

两种方法在求解器中常结合使用，根据问题动态切换以加速求解，求解流程可以概括如图2.4所示。

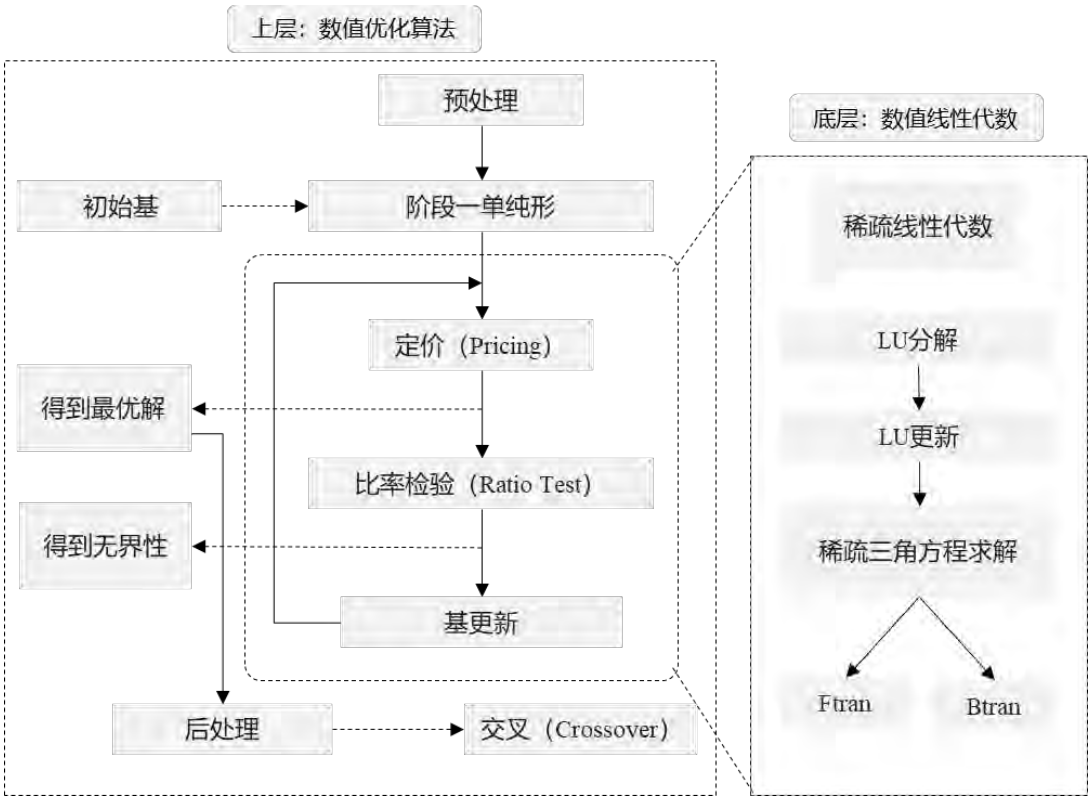

图2.4 线性规划单纯形法算法流程

单纯形求解器的关键模块包括：

1) 初始化：找到初始可行解，如通过两阶段法或者大M法。

2) 迭代步：包括选择出、入基变量，对应pricing和ratio-test两大涉及算法性能和数值稳定性的关键技术，属于求解器的技术机密；每个迭代步也依赖数值线性代数，进行基矩阵的分解和矩阵-向量乘积计算。

3) 终止：判断问题状态，包括无界、不可行、可解，和输出满足精度要求的最优解。

此外，单纯形算法可以拓展到二次规划，即目标函数含有二次项的情景。

（二）内点法

内点法是一类现代算法，通常比单纯形法更快，尤其在处理大规模稀疏问题时[29]。它的原理是通过在可行域内部沿障碍路径（Barrier Path）逼近最优解，避免在边界上移动。常用方法包括投影梯度法、牛顿法等。

不同于单纯形算法对初始点的强依赖，内点法求解器对初始点的依赖相对较弱，甚至不必指定可行的初始点。关键模块是每个迭代步需要求解的增广线性方程组，其求解精度、稳定性关系着整个内点法求解器的性能和鲁棒性。此外，内点法可以扩展至其他一般非线性连续问题求解。

为了方便其他连续优化模型的内点法理论阐述，本白皮书将模型抽象到一般非线性优化（连续）问题：

$$f^* = \min\{\mathrm{f}(x) \mid g_i(x) \le 0, i = 1, \cdots, m, x \in \mathbb{R}^n\}$$

这里，LP问题对应上述模型中 $\mathrm{f}(x) = c^T x$，$g_i(x) = a_i^T x$，其中 a_i 为LP的约束矩阵 A 的行，$i = 1, \cdots, m$。

内点法的收敛性要求可行域严格内点非空，即存在x满足 $g_i(x) < 0$。它首先构造对数障碍函数，对每个不等式约束引入对数障碍项：

$$\varphi(x) = -\sum_{i=1}^{m} log(-g_i(x))$$

1) 当 $g_i(x) \to 0^-$时，$\varphi(x) \to \infty$；

2) 当 $g_i(x)$远离边界时，$\varphi(x)$近似线性。

对障碍参数 $\mu > 0$ 构造障碍问题（又称中心路径问题）：

$$\min\{\mathrm{f}(x) + \mu\varphi(x) \mid x \in \mathbb{R}^n\}$$

然后使用牛顿法迭代逼近最优解，即：

$$\nabla^2(f + \mu\phi)\Delta x = -\nabla(f + \mu\phi)$$

这里梯度的计算公式为：

$$\nabla\big(f(x) + \mu\phi(x)\big) = \nabla f(x) - \mu\sum_{i=0}^{m} \frac{\nabla g_i(x)}{g_i(x)}$$

和海塞矩阵的计算公式为：

$$\nabla^2(f(x)+\mu\phi(x)) = \nabla^2 f(x) - \mu\sum_{i=1}^{m}\left(\frac{\nabla^2 g_i(x)}{g_i(x)} - \frac{\nabla g_i(x)\nabla g_i(x)^{\mathrm{T}}}{g_i(x)^2}\right)$$

牛顿方程组的求解应尽可能利用稀疏性和多线程并行计算，提高求解效率。

依赖于具体的模型定义，如LP中的线性函数的梯度为常数向量而海塞矩阵为零，而非线性模型中的二次函数梯度则为线性函数、海塞矩阵为常数矩阵。图2.5概述了一般数学规划内点法算法流程。当模型涉及的函数满足凸性要求时，内点法可以保证收敛到问题的最优解，否则至局部最优状态。

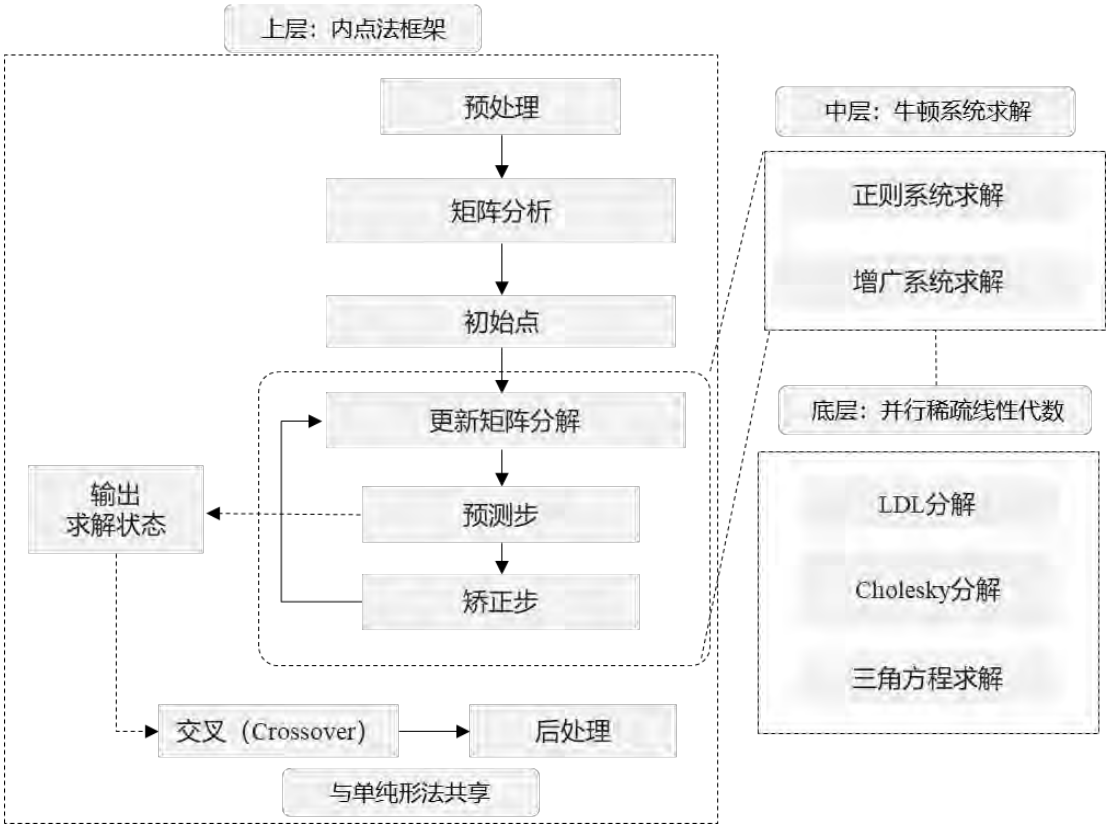

图2.5 数学规划内点法算法流程

特别地，对于LP问题，若使用单纯形方法获取“顶点解”，需要启用交叉步骤（Crossover Steps），切换到单纯形法，利用内点法的近似解作为起点，输出一个精确的顶点解（单纯形法的最优解）。Crossover是连接内点法和单纯形法的桥梁，通过利用两者的互补优势，提升了线性规划求解的效率和鲁棒性。

（三）LP求解器扩展功能

除了精确获取LP模型的求解状态外，易用性强的LP求解器还提供以下功能便于用户调测运筹优化方案。

灵敏度分析（Sensitivity Analysis）分析是LP决策的核心工具，能够量化模型参数的不确定性影响，辅助制定灵活策略。实际应用中需结合求解器功能与数学推导，确保结果可靠性。它主要用于研究模型参数变化对最优解的影响，是预处理和求解后的关键后处理步骤，关注角度包括：

1) 参数变化时，当前最优基是否仍最优；

2) 在不改变最优基的前提下，参数允许的变化范围是多少。

不可行诊断功能（Infeasibility Diagnosis）是用于识别并解释模型为何无可行解的技术。当线性规划问题无可行解时，求解器会通过诊断工具帮助用户定位导致矛盾的约束或变量，从而快速修复模型，包括：

1) 定位矛盾约束：找出导致不可行的极小约束集合，即不可约不可行子系统（Irreducible Infeasible Subsystem，IIS）；

2) 提供修复建议：通过调整约束或变量，使模型恢复可行性，修复建议包括删除或修改约束，或将硬约束转为软约束从而重新定义优化目标。

通过 IIS 检测、冲突分析和松弛变量法，帮助用户快速定位矛盾约束或得到可行松弛模型。

2.1.1.6 二次约束二次规划关键技术

二次约束二次规划是一类重要的非线性优化问题，其目标函数或者约束函数中含有二次项。它广泛应用于金融、工程、机器学习等领域[30]。它的数学标准形式为：

$$\begin{aligned} \min\ & \frac{1}{2}x^T Q_0 x + {c_0}^T x \\ s.t.\ & x^T Q_i x + {c_i}^T x \le b_i, i = 1, \cdots, m \\ & x \in R^n \end{aligned}$$

其中，x 为决策变量向量，$Q_0, Q_1, \cdots, Q_m$ 为对称矩阵，$c_0, c_1, \cdots, c_m$ 为线性系数，$b_1, \cdots, b_m$ 为资源限制向量。它包括了二次规划（Quadratic Programming，QP）和二次约束规划（Quadratically Constrained Programming，QCP）作为特例，其中QP对应$Q_1, \cdots, Q_m$ 为零即所有约束函数为线性的情形，而QCP对应 Q_0 为零，即目标函数为线性的情形。

此外，若所有的矩阵都是半正定矩阵（对于所有的非零向量 $x \in R^n$，满足 $x^T Q_i x \geq 0$，$i = 0, \cdots, m$），则QCQP为凸问题，否则为非凸问题，可能有多局部极值，需要全局优化算法，依赖于启发式算法、凸松弛或者求解一系列近似二次规划问题。

这里，本白皮书重点讨论凸QCQP规划算法技术。它的原理类似LP的内点法，通过引入障碍函数（Barrier Function）将不等式约束转化为惩罚项，使用牛顿法迭代逼近最优解。这里，二次函数的梯度和海塞矩阵表达式显式地利用了矩阵的信息，其中梯度为线性函数，海塞矩阵为常数。算法流程核心模块与LP内点法一样，关键在于增广线性方程组的稳定、高效求解，除此之外，关键技术点还包括：

1) 预处理技术：实际应用中不可或缺的步骤。消除冗余约束，简化问题规模，减少计算复杂度；问题标准化与变量缩放可以减少数值误差，提升数值稳定性；

2) 初始点的选择：对内点法的收敛速度和稳定性也是非常重要的。首先，初始点必须严格满足所有不等式约束（即严格位于可行域内部），避免算法在边界上发散，其次，在原始-对偶内点法中，初始点的对偶变量（拉格朗日乘子）需要满足正定性条件，再次，初始点的坐标不能过大或过小，避免数值计算中的溢出或精度问题；

3) 对数障碍参数的选择：一般需要结合迭代步数和中心参数，以合适的节奏趋向于零，对收敛性的影响较显著。

2.1.1.7 二阶锥规划关键技术

二阶锥规划是优化问题中的一类重要凸优化问题，其目标函数和约束条件涉及二阶锥（Second-Order Cone，SOC）结构。SOCP在工程设计、金融优化、机器学习等领域有广泛应用[31]。

二阶锥是在 k 维空间中定义的一种特殊集合，由欧几里得二范数定义。其标准形式为 $SOC = \{(x, t) \in \mathcal{R}^{k-1} \times \mathcal{R}: \| x \|_2 \leq t\}$

1) 在三维空间中，SOC的几何形状是一个双曲锥（Hyperboloid），其边界满足 $\sqrt{x_1^2 + x_2^2} \leq t$；

2) 在四维空间中，SOC的几何形状是一个旋转对称的圆锥，其边界满足

$\sqrt{x_1^2 + x_2^2 + x_3^2} \leq t$。

注意，二阶锥是一种凸锥，即集合内任意两点的线性组合（且系数非负）仍在集合内。

二阶锥约束，又称为范数约束，它通常表示为 $\| Ax + b \|_2 \leq c^T x + d$，涵盖了线性规划、二次规划等，在工程和金融优化中非常高效。相应地，SOCP数学模型表达式为：

$$c^* = \min \{ c^T x \mid \| A_i \mathrm{x} + b_i \|_2 \leq c_i^T x + d_i, i = 1, \cdots, m, x \in \mathbb{R}^n \}$$

内点法是求解SOCP最高效的算法之一，不同于分解方法或近似方法（如交替方向乘子法，Alternating Direction Method of Multipliers，ADMM），内点法为多项式算法，在中等规模问题中表现优异，可以较快地收敛到高精度解。在SOCP的内点法求解框架中，每个锥约束的障碍函数为：

$$\phi(x) = -log(c_i^T x + d_i - \| A_i \mathrm{x} + b_i \|_2)$$

总障碍函数为所有锥约束的障碍函数之和，用于构造中心路径，即通过引入障碍参数 μ 将原问题转化为无约束问题：

$$\min \left\{ c^T x - \mu \sum_{i=0}^{m} log(c_i^T x + d_i - \| A_i \mathrm{x} + b_i \|_2) \mid x \in \mathbb{R}^n \right\}$$

中心路径随着 $\mu \to 0$ 收敛到原问题的最优解。每个迭代步通过牛顿法迭代逼近最优解，同时保持严格可行性。因而牛顿系统的高效和稳定求解仍是SOCP计算的关键。

对比LP、QCQP和其他非线性规划问题，SOCP的KKT矩阵通常具有块对角结构（每个锥对应一个块），可分解为多个小规模子问题，降低计算复杂度。其他关键技术点包括：

1) 预处理技术，提升数值稳定性：移除冗余约束或合并相似约束，归一化变量和约束系数等；

2) 稀疏性利用：对稀疏矩阵，包括计算过程中涉及到的雅可比和海塞矩阵采用稀疏存储格式（如CSR）减少内存占用和计算量；

3) 并行化策略：采用多线程并行处理矩阵的相关计算；

4) 对数障碍参数μ的选择：一般需要结合迭代步数和中心参数，以合适的节奏趋向于零，对收敛性的影响较显著；

5) 预测-校正方法：通过预测步长和校正步长平衡收敛速度与稳定性。

2.1.1.8 一般非线性规划关键技术

非线性系统通常能更精准地描述设计需求，却局限于求解复杂和困难，是金融、电网、化工、制造业等领域中的很多优化决策问题面临的挑战之一[32][33][34]。一般非线性规划的数学描述如下：

$$\begin{aligned} & \min \ f(x) \\ s.t. \quad & g_L \le g(x) \le g_U, \\ & x_L \le x \le x_U, \\ & x \in R^n. \end{aligned}$$

这里，约束或目标函数可以是一般非线性类型。

根据约束和目标函数性质，NLP可以分为凸、非凸两类问题。当模型具备凸性时，所收敛到的极值即为模型的全局最优解；否则需要依赖分支定界方法寻找全局最优。一般非线性求解器使用内点法求解模型的局部最优，在某个邻域范围内是最优的，但不一定是全局最优解。

NLP不像LP、QCQP及SOCP那样导数信息可以轻易地显式获取，在用内点法求解时，需要提前给出NLP的雅可比和海塞公式。内点法的策略模块，如预处理、对数障碍参数的选择、稀疏并行计算也是NLP内点求解器的性能关键。

2.1.1.9 混合整数线性规划关键技术

混合整数线性规划求解器致力于求解以下混合整数线性规划问题：

$$c^* = \min\{c^T x \mid Ax \le b, x \in \mathbb{R}^n, x_j \in \mathbb{Z}, \forall j \in I\}$$

其中 $X_{MIP} = \{x \in \mathbb{R}^n \mid Ax \le b, x_j \in \mathbb{Z}, \forall j \in I\}$ 称为MILP的可行解集合，满足 $x^* \in X_{MIP}$ 和 $c^T x^* = c^*$ 的解称为最优解。此外，求解器会处理简单边界的约束 $l_j \le x_j \le u_j, u_j, l_j \in R \cup \{\pm\infty\}$。MILP问题对应的LP松弛问题：

$$c^{\vee} = \min\{c^T x \mid Ax \le b, x \in \mathbb{R}^n\}$$

其中LP问题的最优目标值满足 $c^{\vee} \le c^*$，因此可作为MILP问题的下界，也称对偶

界。利用求解LP松弛问题的分支-定界或分支-割平面方法是MILP问题的通用求解方法[35][36]。通过输入特定问题文件（如MPS、LP格式文件），用户可调用求解器对问题进行求解，并得到问题的可行性和最优性信息。

由于一般的MILP问题具有NP-难属性，基本的分支-割平面方法难以满足大规模问题的求解需求。为了极大化问题求解能力，主流的MILP求解器在基本的求解框架中集成了多个可加速求解的算法模块，如预处理、域传播、启发式、冲突分析、对称性检测等，并囊括了业界绝大多数优秀的解决方案。除基本的串行求解能力外，多数商业求解器还具备单机并行和多机分布式并行求解能力，可充分利用硬件资源应对更复杂的应用场景。针对求解效率优先和可靠性优先两个目标，求解器分别提供了非确定性和确定性并行功能，实现差异需求的覆盖。作为一款高效、易用的数学软件，MILP求解器的通用模块兼顾高效执行和用户友好两个指标。求解器既提供了通用问题文件读取功能，也提供基本的建模能力，降低了使用门槛。通过命令行参数或传入JSON文件等方式，用户可根据自身需求对求解器算法参数进行自定义，满足定制化求解需求。

接下来详细介绍主流混合整数规划求解器的架构模块和主要算法模块。

（一）架构模块

主流MILP求解器架构模块分为用户交互接口、算法层、算法与功能管理模块、主管理类、分支-割平面框架等五个层次。各模块的功能说明如表2.1所示。

表2.1 MILP求解器架构模块功能说明

层次	模块	功能
用户交互接口	Command Line	命令行入口
	Interactive Shell	交互式Shell入口
	C++/JAVA/Cloud API	C++/JAVA/云服务调用接口
算法层	Presolves	预处理算法，主要功能为降低问题规模
	Heurs	启发式算法，主要功能为快速找到原始可行解
	Propers	域传播算法，主要功能为快速缩减可行域
	Sepas	割平面算法，主要功能为缩减可行域
	Conflict	冲突分析算法，主要功能为剪枝、缩减可行域
	Branch&NodeSelect	分支和节点选择算法，分支定界算法的基础
	Readers	文件读取功能

算法与功能管理模块	PresolManager	预处理算法管理和调度
	HeurManager	启发式算法管理和调度
	PropManager	域传播算法管理和调度
	SepaManager	割平面算法管理和调度
	ParamManager	参数管理
	StatManager	统计信息管理
主管理类	Solver/Coordinator	串、并行控制
分支-割平面框架	Prob	问题数据
	Cons	约束数据
	Vars	变量数据
	Tree	分支定界树
	CutPool	割平面池
	Sol	原始可行解
	Conflicts	冲突约束集合
	Cliques	团集合
	LP	LP数据及LP求解器接口

图2.6展示了串行模式下的MILP分支-割平面算法流程，其中包括：预处理、节点选择、节点计算、域传播、割平面、约束强化和原始启发式等几个阶段或算法环节。本白皮书将分别对这些环节进行简述。

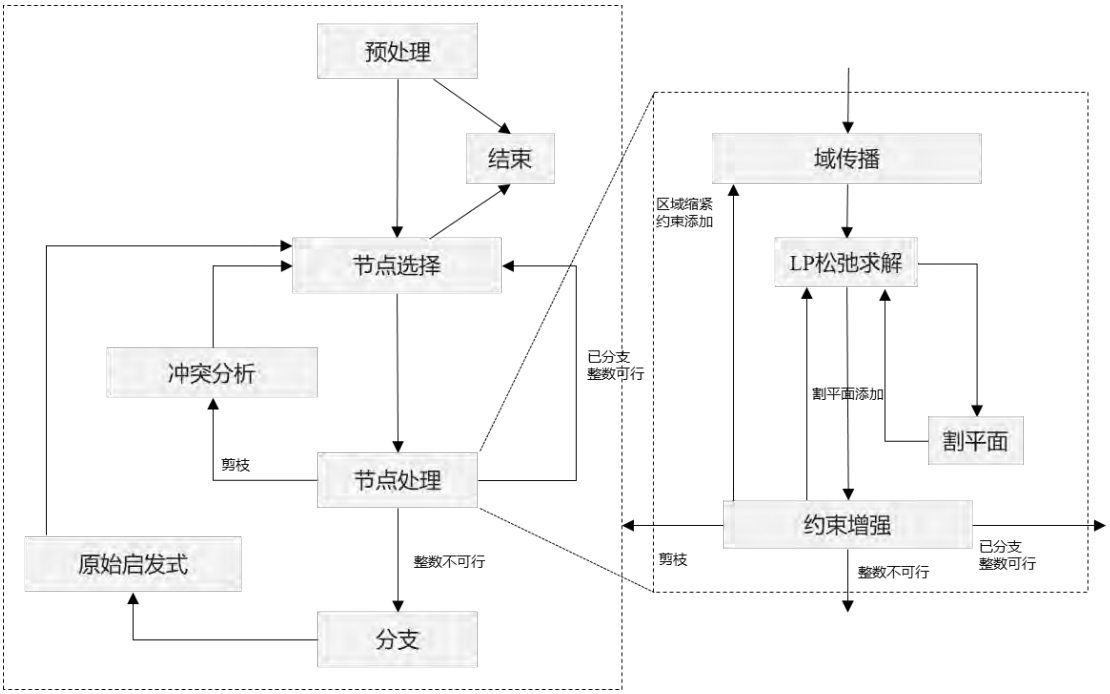

图2.6 混合整数线性规划分支-割平面算法流程

（二）主要算法模块简介

(1) 预处理

在预处理阶段会反复调用预处理插件中集成的预处理算法，直到问题无法再进行进一步约减，或终止条件被触发[37]。预处理阶段对问题的改变是永久性的，因此预处理通常只在分支定界树的根节点进行。在分支节点只会进行有限的预处理，即缩减变量的界，称为域传播。

预处理阶段的主要目的是对原问题的变量和约束进行约减，降低问题的求解难度。预处理阶段还会完成特定类型约束的识别、转换和处理工作[38]。

(2) 节点选择

在MILP求解过程中，每次迭代首先要确定的就是当前要处理的分支节点。所选择的节点可以是当前分支的子节点，或者其他分支上优先级最高的节点。节点优先级的计算和节点的选择由节点选择模块负责[39]。

连续地选择当前分支的子节点被称为潜水（Plunging）。由于子节点可利用父节点的信息（如LP问题的最优基），因此求解器可以更高效的执行潜水，这个过程被称为热启动（Warm Start）。潜水实际上就是执行深度优先搜索（Depth First Search，DFS），与之相对的是最佳优先搜索（Best First Search, BFS），即每次都选择优先级最高的节点进行处理。使用深度优先策略的优点是可以更快的找到可行解，并且深度优先的方法需要的内存更小；最佳优先搜索则在改进对偶界（Dual Bound）方面更有优势[40]。

(3) 原始启发式

原始启发式算法[41][42]可分为初始型启发式算法（Starting）和改进型启发式算法（Improvement），其中初始型启发式算法的目的是快速找到MILP问题的整数可行解，这类算法的计算代价通常较小；改进型启发式算法的作用是在LP松弛问题的最优解和当前已找到的可行解的基础上搜索目标函数值更小的解，从而改善原始界（Primal Bound）。改进型算法的计算通常更为复杂。

主流MILP求解器会在运行的不同阶段调用原始启发式方法，具体的调用阶段有：

1) 在求解LP松弛问题之前调用的原始启发式方法被称为Before-LP，即不需要LP问题的解就可以找到MILP问题的可行整数解；

2) 在节点处理阶段调用的原始启发式算法，通常需要基于LP问题的最优解进行改进；

3) 在完成某个节点的处理后进行调用，此时调用的通常是改进型原始启发式算法。

(4) LP松弛

求解LP松弛问题可以获得当前分支的对偶界。LP问题的最优解也可以为接下来的搜索提供指导。主流MILP求解器使用单纯形类方法或内点法对LP松弛问题进行求解。由于单纯形法可以利用从父节点继承的最优基的信息，因此可以实现LP求解的热启动；基于相同的原因，内点法在求解LP子问题时的求解效率通常不及单纯形类算法，但在特定问题上内点法具有明显的优势。

(5) 割平面

在LP问题完成求解之后，MILP求解器会调用割平面算法来收紧松弛问题[43][44]。在完成一次LP求解后会生成多个割平面，但只会将部分割平面会作为新的约束添加到LP问题中。与LP问题违反度高（距离LP问题最优解距离远），与已添加的割平面相似度低（正交）的割平面会被优先添加。一些割平面可能只会改变变量的界，这时MILP求解器会调用域传播模块进一步缩小变量的取值范围。

(6) 约束增强

在完成一轮LP求解和割平面添加后会对当前的求解状态进行分析，决定下一步要执行的操作，这一过程被称之为约束强化。这一阶段可进行的操作有：

1) 如果判断当前分支不可行，或已找到当前分支的最优解，则说明已完成对当前分支的搜索，直接进入节点选择阶段；

2) 进一步收紧变量的界或增加新的约束，执行域传播算法，重启一轮松弛-割平面运算；

3) 如果当前分支需要被剪除（不可行或局部对偶界超过全局原始界），则进入冲突分析阶段，尝试利用当前的分支的信息剪除其他分支；

4) 添加新的割平面，重新求解LP问题；

5) 对当前节点进行分支，派生出新的子问题。

2.1.1.10 混合整数二次规划关键技术

混合整数二次规划[45][46]是混合整数规划（Mixed Integer Programming，MIP）的一种特殊形式，其目标函数为二次型且部分变量为整数，通常表示为：

$$\min\left\{\frac{1}{2}x^T\mathrm{Q}x+\ c^Tx \mid Ax \le b, x \in \mathbb{R}^n, x_j \in \mathbb{Z}, \forall j \in I\right\}$$

这里，Q 是对阵矩阵。MIQP广泛应用于金融优化、调度、机器学习（如支持向量机）、控制工程等领域。根据目标函数是否具有凸性，MIQP可以分为两类：若 Q 半正定，该模型为凸MIQP，否则为非凸MIQP。

与MILP和MIQP密切相关的另外一类问题是混合整数二次约束规划（Mixed Integer Quadratically Constrained Programming, 简称MIQCP），它的目标函数一般是线性的，但约束中包含二次项，一般形式为：

$$\min\{c^Tx \mid \ x^TQ_ix+\ {c_i}^Tx \le b_i, i=1,\cdots,m, x \in \mathbb{R}^n, x_j \in \mathbb{Z}, \forall j \in I\}$$

这里 $Q_1,\cdots,Q_m$ 为对称矩阵，若它们为半正定矩阵，则上述模型为凸MIQCP问题；否则为非凸MIQCP问题。

这两类问题通常被一起提及，其求解复杂度较高，属于NP难问题，MIQP和MIQCP的求解算法大概分为以下几类：

1) 确定性算法，如分支定界（Branch and Bound, 简称B&B）递归划分变量空间（分支）并求解连续松弛问题（定界），割平面法通过添加线性约束（割平面）逐步逼近整数解；

2) 启发式算法，如遗传算法，模拟退火，禁忌搜索等；

3) 结合确定性算法和启发式算法的混合算法，如分支与割平面法结合分支定界与割平面法，在分支过程中动态添加割平面，和分解方法将问题分解为主问题（整数变量）和子问题（连续变量）、适用于变量规模较大的问题。

凸MIQP的求解方法通常结合分支定界和二次规划技术。在分支定界过程中，每个节点求解连续二次规划松弛，利用对偶理论和割平面加速收敛。算法流程如图2.7所示。

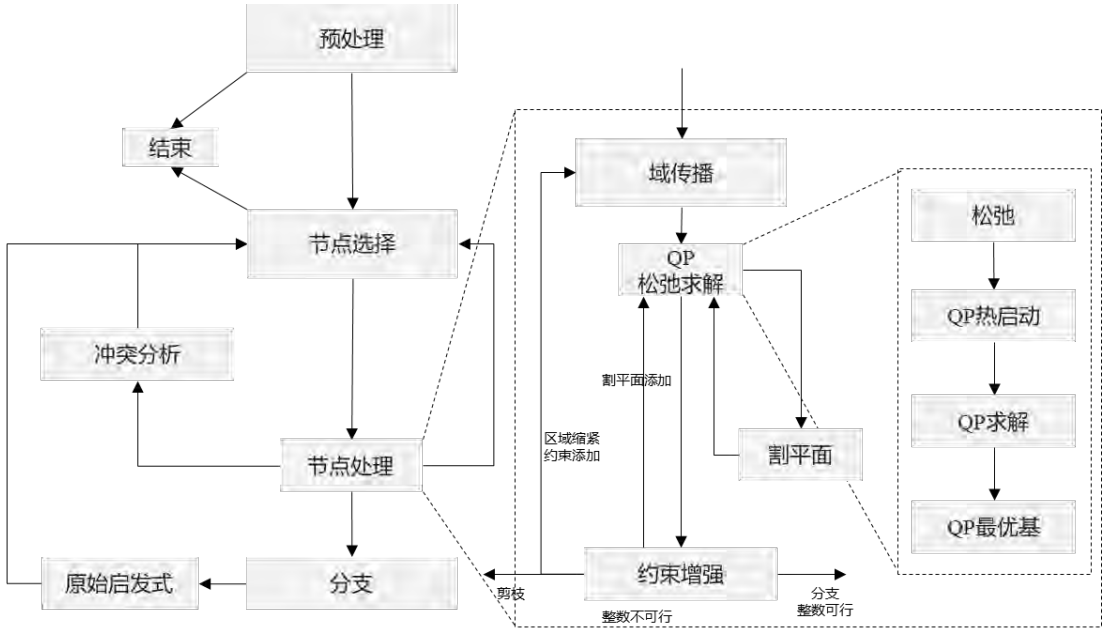

图2.7 混合整数二次规划求解算法流程

MIQP的主要算法模块分布在三个方向：

1) 预处理模块：对原问题的变量和约束进行约减，降低问题的求解难度，包括通过边界分析固定部分变量和线性化某些二次项；

2) 分支定界算法模块：类似于MILP的求解流程，结点选择、原始启发式、分支策略、域传播，冲突分析和域传播，甚至并行和分布式计算，都对MIQP的搜索求解有着重要影响；

3) 节点/松弛问题的求解：MIQP的松弛问题为QP问题，在实际求解过程中，QP问题的单纯形求解器往往被调用，得益于其便利的热启动功能。

类似地，凸MIQCP的求解方法也通常结合分支定界和二次约束规划技术，每个分支定界节点求解连续的二次约束问题（通过调用QCP或者SOCP求解器）。具体算法流程如图2.8所示。

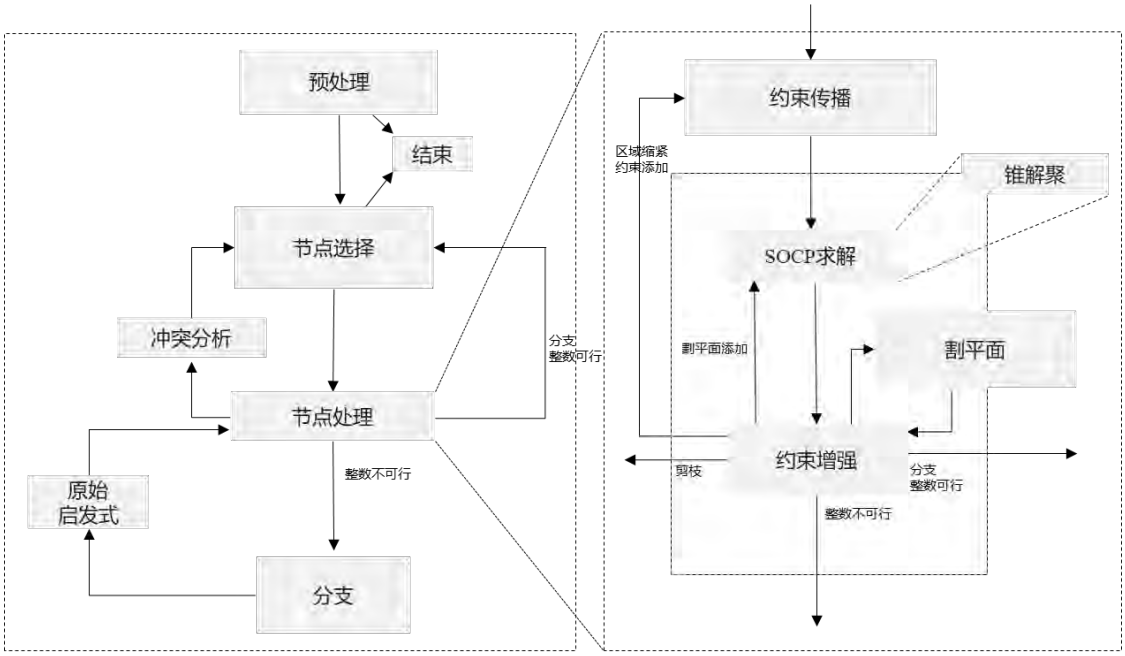

图2.8 混合整数二次约束规划求解流程

与凸MIQP求解器一样，MIQCP求解器也受预处理和分支定界各个模块的影响。用于求解子问题的连续求解器精度和性能对整体收敛速度非常重要的。

综上所述，MIQP和MIQCP和MILP求解器共同依赖分支定界组件和预处理，该部分可以抽象成共享模块，各个求解器在此基础上进行独有的算法特色化。

2.1.2 约束规划求解器技术

2.1.2.1 约束规划简介

约束规划（Constraint Programming，CP）[47][48]是一类用于求解组合优化与离散决策问题的通用方法。它通过变量（Variables）、取值域（Domains）、约束（Constraints）和目标函数（Objective）来描述问题，并利用系统化的搜索与推理机制寻找满足所有约束的最优解或可行解。约束规划广泛应用于各种组合优化问题，例如排程、资源分配、路径规划、配置问题等。与传统的数学规划方法（如线性规划、整数规划）不同，约束规划更注重对问题的逻辑建模，而非严格的数学表达，因此在处理复杂约束场景时具有显著优势。

一个典型的约束规划模型可以形式化为：

变量与域：

$$X = \{x_1, x_2, \dots, x_n\}, \qquad \forall i \in \{1, \dots, n\}, x_i \in D_i$$

约束：

$$C = \{c_1, c_2, \dots, c_m\}$$

目标函数：

$$\min/\max f(X)$$

约束规划求解器的代表性技术路线有三类。第一类是传统的约束规划精确求解器，如IBM的CP Optimizer，它的核心方法是基于域传播、树搜索、全局约束推理等经典的约束规划精确求解方法，通过变量域收缩、搜索树剪枝等方法寻找全局最优解；第二类是非完备的启发式算法求解器，如Hexaly，它的核心方法是大邻域搜索、局部搜索等启发式算法，不保证最优性，但能够快速找到高质量可行解；第三类是CP-SAT混合求解器[49][50]，如Google的OR-Tools，它的核心方法是基于懒惰子句生成方法利用先进的SAT（布尔可满足性问题）[51]求解技术解决约束规划问题，OR-Tools中融合了约束规划、SAT、线性规划、混合整数规划等多领域的技术，同时也融合了大邻域搜索、局部搜索等多种启发式算法，求解能力强，是当前性能最好、技术覆盖面最全的约束规划求解框架。

2.1.2.2 约束规划求解器软件架构

约束规划求解器的架构被划分为四个主要层次：应用层、算法层、工具层和系统层，如图2.9所示。每一层都有其特定的功能和职责，共同确保求解器的高效运行和良好的用户体验。

图2.9 约束规划求解器软件架构

应用层是用户与求解器交互的最外层，主要负责处理外部请求和数据交换。它包括第三方接口，允许与其他系统或服务无缝集成，支持多种编程语言的用户API，如Python、Java、C++等，以满足不同开发环境的需求。此外，应用层还包含标准数据格式的输入输出（I/O）功能，如MPS（Mathematical Programming System）和FZN（FlatZinc），这些格式在优化问题建模中广泛应用，确保数据的兼容性和可移植性。通过这些接口和数据格式，用户可以方便地将问题模型导入求解器，并将结果导出到其他系统中进行进一步处理。

算法层是求解器的核心，包含了多种关键算法和技术，用于高效地解决约束规划问题。首先，预处理技术用于简化问题，通过分析和消除冗余约束，减少搜索空间，从而提高求解效率。其次，基于SAT（布尔可满足性问题）的迭代算法，通过将约束问题转化为布尔逻辑问题，利用高效的SAT求解器进行求解，特别适用于处理复杂的逻辑约束。懒惰子句生成算法则通过动态生成和管理子句，进一步优化搜索过程，避免不必要的计算。启发式算法用于在大规模问题中快速找到近似最优解。此外，并行优化技术利用多核处理器的并行计算能力，加速求解过程，显著提升处理复杂问题的效率。

工具层提供了实现算法所需的基础库和工具，为上层算法的实现提供了坚实的技术支持，确保算法的高效性和可靠性。

系统层是整个架构的基础设施，负责管理和协调底层资源。内存管理模块优化

内存使用，减少内存碎片和分配开销，提升系统性能。线程管理模块协调多线程任务的执行，确保并行计算的高效性和稳定性。文档管理模块提供详细的文档和帮助信息，方便用户理解和使用系统。测试模块通过自动化测试和验证，确保系统的稳定性和可靠性。日志与回调机制记录系统运行过程中的关键信息，帮助用户跟踪和调试问题，同时允许用户在特定事件发生时自定义回调函数，增强系统的灵活性和可扩展性。

通过这四个层次的协同工作，约束规划求解器能够高效地处理各种复杂的约束问题，为用户提供强大而灵活的解决方案。

2.1.2.3 约束规划求解器求解流程

约束规划求解器的整体求解框架如图2.10所示。在预处理之后，对各种完备策略和非完备策略进行并行调度。其中完备策略主要为基于SAT的迭代算法，其核心思想为将优化问题转化为一系列判定问题，并迭代求解，具体细节将在后文介绍。通过参数调整与策略选择，基于SAT的迭代算法可衍生出多种不同的子求解器。此外还有一些其它的基于树搜索框架的完备子求解器。非完备策略即为各种启发式算法，包括局部搜索、大邻域搜索等，详见后文。完备算法与非完备算法将在同步点进行交互，共享解、生成的新约束、缩紧的目标函数上下界、缩紧的全局变量域等等。同步后若发现目标函数上下界一致，或达到指定的最优间隙限制，或发现问题不可行，则求解结束，否则进行下一轮的并行调度。

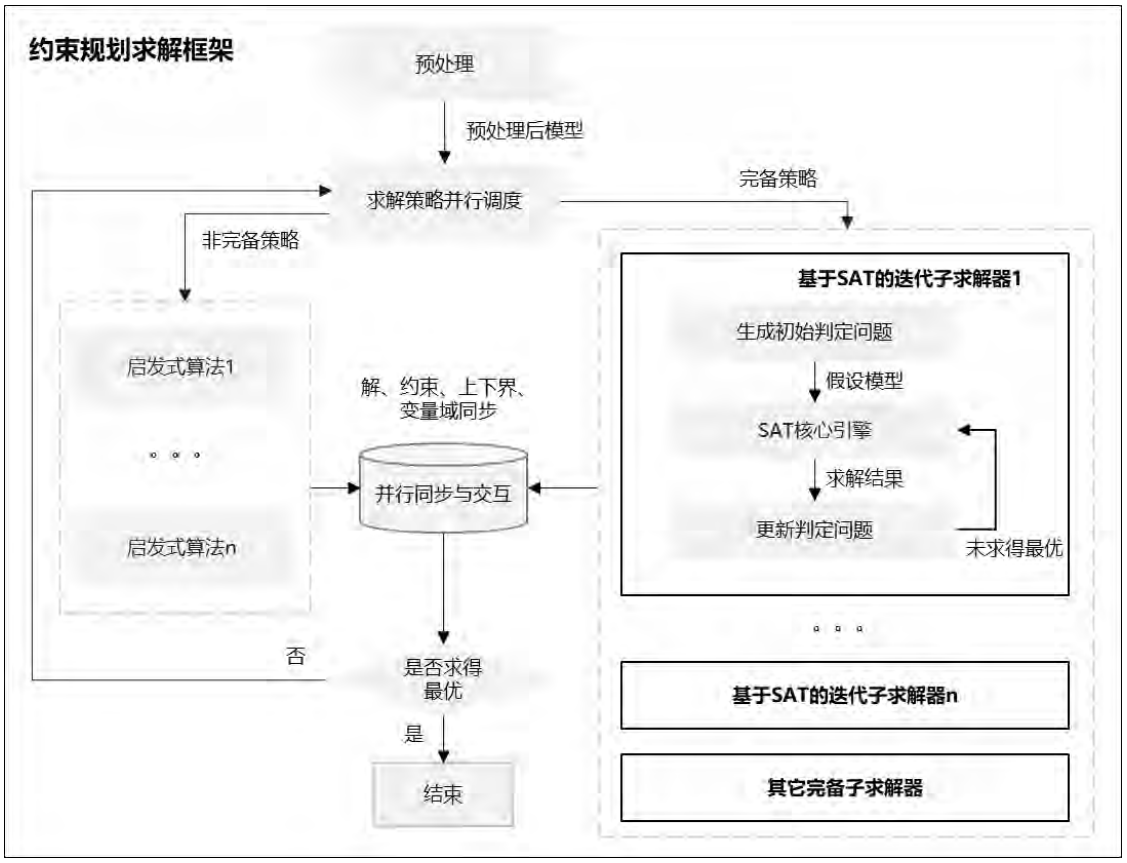

图2.10 约束规划求解流程

2.1.2.4 约束规划求解器功能特性

约束规划求解器作为用户解决约束规划问题最优解搜索的工具，应提供基础的建模、参数设定、求解、结果获取等功能，也应提供回调、模型不可行性分析等高阶功能。主流约束规划求解器应支持如下所述功能。

（一）模型支持

应支持主流约束规划的建模和求解，支持布尔变量、整数变量等变量类型的任意组合，支持任意整数区间的整数变量定义，支持布尔与、或、异或、蕴含、至多/仅有一个布尔变量为真等逻辑约束，支持任意整数区间的线性约束，支持互异约束、表约束、元素约束、互逆约束等全局约束，支持区间约束、（2D）无重叠约束、累积约束、资源池约束、环路约束、路径约束等特定问题约束，支持由整数表达式取最小值、最大值、绝对值、乘法、除法、取余等操作所组成的计算约束，支持多个目标函数，支持求解最大化或最小化问题，支持求解不可行问题。约束规划求解器建模地图如图2.11所示。

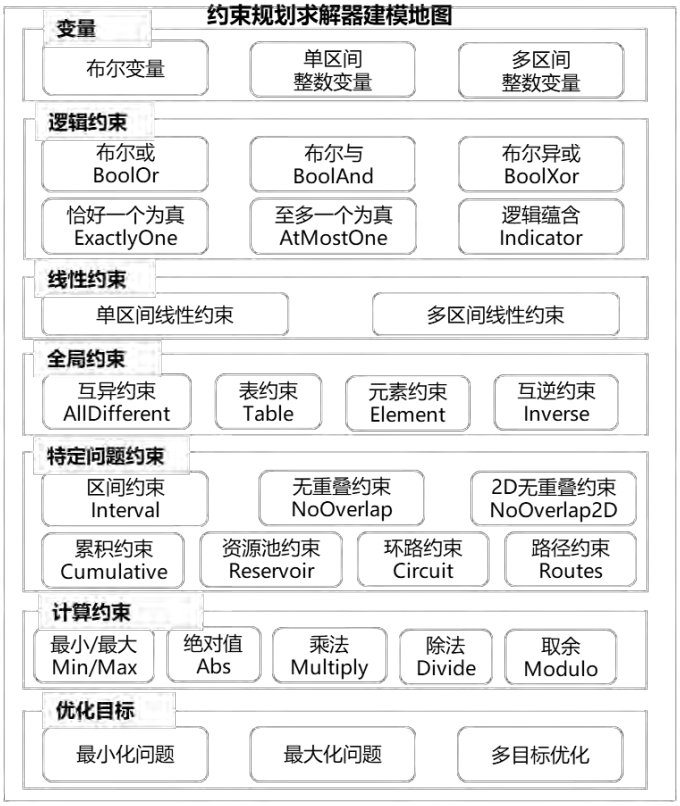

图2.11 约束规划求解器建模地图

（二）求解能力

应支持精确求解和启发式求解两种模式，精确求解器内置预处理、域传播、树搜索、启发式等约束规划求解器常用功能模块，在精确求解模式下，确保在资源允许范围内给出可证最优解，或证明问题的不可行性。

（三）数据输入

应支持约束规划常用模型文件（如FZN及其压缩格式），及整数线性规划问题常用模型文件（如MPS、LP及其压缩格式）读写功能。

（四）日志和结果输出

应支持求解信息打印，包括输入模型简要信息、预处理后模型简要信息、使用的线程数、求解进度（如当前最优上下界、相对最优间隙、当前求解时间），求解结果汇总（如总时间、求解状态、最优上下界、相对最优间隙）。

（五）参数设置

应支持求解时间上限、最大可用线程数、相对/绝对最优间隙容忍度、随机种子等常用参数设置；应支持多种模型预处理、启发式、SAT求解、迭代搜索策略，

并可通过参数进行开关或程度控制。

（六）功能接口

应支持预处理后模型导出、初始解设置、分支变量优先级设置、回调函数等功能；应支持中断求解，并返回中断时的结果。

（七）并行加速

应支持多线程以及多节点分布式求解，应支持确定性并行求解。

2.1.2.5 约束规划关键技术

约束规划求解器中的关键技术包括整体求解流程、预处理、基于SAT的迭代算法、懒惰子句生成、启发式算法、数学规划技术融合、并行优化等模块。

（一）预处理

预处理模块主要用于在正式求解或计算之前，对输入数据或模型进行清洗、简化、转换或结构化，以缩小后续算法的搜索空间，提升搜索效率、稳定性和质量。约束规划求解器的预处理算法地图如图2.12所示，这些算法可分为三大类：

图2.12 约束规划求解器预处理算法地图

1) 特定约束处理。针对某类约束设计的特定处理算法。其中最具代表性的是线性约束和SAT约束（合取范式子句），因为这两类是约束规划问题中的核心约束，且算法种类最多，相关研究最成熟，许多方法可以从数学规划和SAT领域借鉴。此外还有针对其它逻辑、全局、特定约束的预处理算法，例如布尔与约束、互异约束、区间约束等；

2) 特殊结构处理。针对模型中存在的特殊结构进行检测与处理的算法。典型的特殊结构包括对称性结构（例如对于形如 $\sum_{i\in\{1,\dots,n\}} x_i \le k$ 的基数约束而言，其中的一系列变量 x_i 都是对称的），以及Clique团结构、AtMostOne结构等等；

3) 变量约束清理。对模型中的变量约束进行域缩紧和清理操作。普通的变量、约束域缩紧是最常见的预处理方法。Probing探测技术是一种特殊的基于向前看思想的算法，这类方法会尝试赋值或限制某个变量，观察赋值后对问题的影响，若赋值后导致不可行则表示此前的赋值假设不成立，可基于此结论对变量的域进行限制，缩小搜索空间。

（二）基于SAT的迭代算法

基于SAT的迭代算法的核心作用在于将优化问题转化为一系列判定问题进行求解，其算法框架如图2.13所示。该方法有效性的前提条件是判定问题的求解方法成熟且高效。这类算法可分为迭代优化上界和迭代优化下界两类，它们针对优化问题构造一系列形如 $Obj \le K$ （优化上界）或 $Obj \ge K$ （优化下界）的问题，而后判断问题是否成立，根据结果可知是否已经求得最优解，若未求得则尝试继续构造问题，进一步逼紧上界或下界。当前判定问题的求解主要采用基于CDCL[51][52]（冲突驱动学习子句）算法框架的SAT求解器进行求解（如图2.13右半部分所示）。

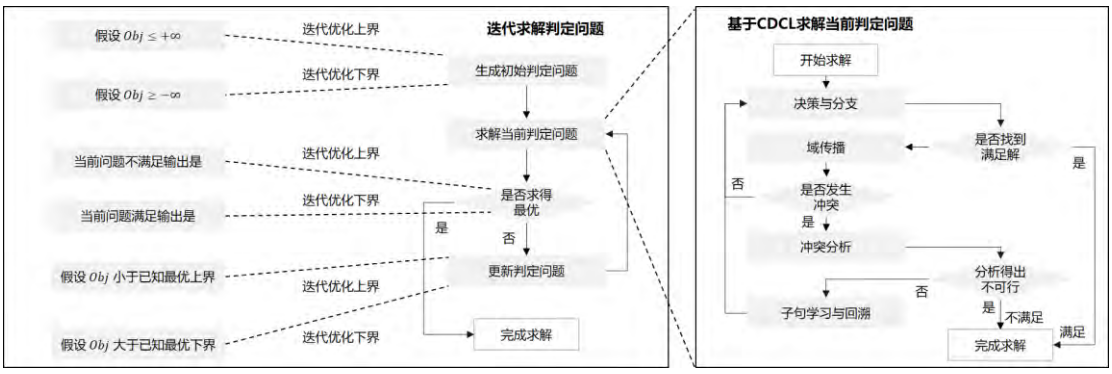

图2.13 基于SAT的迭代算法流程

约束规划求解器中常用的SAT迭代算法有三种：

1) 线性扫描是迭代优化上界的一种算法。首次求解时忽略目标函数，仅考虑约束条件，此时若求解得到不可行则问题证明为不可行，否则可得到一个可行解，假设其目标值为 K，接下来向模型中添加 $Obj \le K-1$ 的约束（默认最小化问题），对问题进行重新求解，若找到可行解则更新约束并继续循环操作，未找到可行解则

说明上次找到的可行解就是最优解；

2) 核引导是迭代优化下界的一种算法。首次求解时对目标函数进行最贪心的尝试，此时若求解得到可行解则该解一定是最优解，否则可基于SAT求解器得到一个不可满足核，接下来对该不可满足核进行最小程度的松弛，而后继续尝试求解，若求得可行解则为最优解，否则将产生新的不可满足核，而后对上述过程循环操作；

3) 最大命中集是迭代优化下界的一种算法。其核心思想是通过逐步扩展一个命中集，即满足某些关键约束的变量组合，来提升问题的下界。算法从一个初始的空命中集开始，逐步添加变量或约束，确保每次扩展后仍保持问题的可行性。通过这种方式，算法能够逐步逼近最优解，并在每一步中验证当前命中集是否能够满足所有约束条件。如果命中集无法进一步扩展，则当前解即为最优解。

（三）懒惰子句生成

懒惰子句生成是一种结合约束规划与SAT求解优势的混合算法，它通过在搜索过程中按需生成子句来增强传统约束规划的推理能力。其核心思想是：约束规划求解器在进行约束传播时，不仅缩小变量域，还会记录导致传播的推理原因，并将这些原因转化为SAT子句。当搜索遇到冲突时，懒惰子句生成会像现代SAT求解器一样执行冲突分析，从推理链中提取一个学习子句，并将其加入约束集中，以避免未来重复进入同样的冲突区域。与SAT的CDCL不同，懒惰子句生成的变量是整数变量，因此它需要将传播事件映射为布尔层面的解释，这使得学习子句能够跨越约束规划的高层语义与SAT的低层推理之间的鸿沟。懒惰子句生成的优势在于，它既保留了约束规划对全局约束和复杂结构的强大传播能力，又继承了SAT求解器在冲突学习和搜索剪枝方面的高效性，从而显著提升求解性能，特别适用于调度、分配、组合优化等复杂离散问题。通过这种懒惰生成子句的方式，系统只在必要时引入新的逻辑约束，使求解过程既精确又高效。

约束规划求解器中的SAT核心引擎框架如图2.14所示，整体可分为六大模块。

1) 预处理与嵌入式处理。在求解前和求解过程中对模型进行调整，加速求解过程。其中经典的算法有很多，包括变量消除、子句包含、子句简化、阻塞子句消除、覆盖子句消除、失败文字检测、传递规约、超三元归结等等；

2) 搜索引导启发式。在求解过程中对搜索进行一些启发式的引导。包括子句管理，即管理与维护学习子句，当学习子句积累过多时删除一些低质量的子句；重

启，即定期回溯到根节点重新搜索；重定向：为变量的赋值提供新的建议；局部搜索：通过局部搜索算法寻找可行解或提供重定向建议；

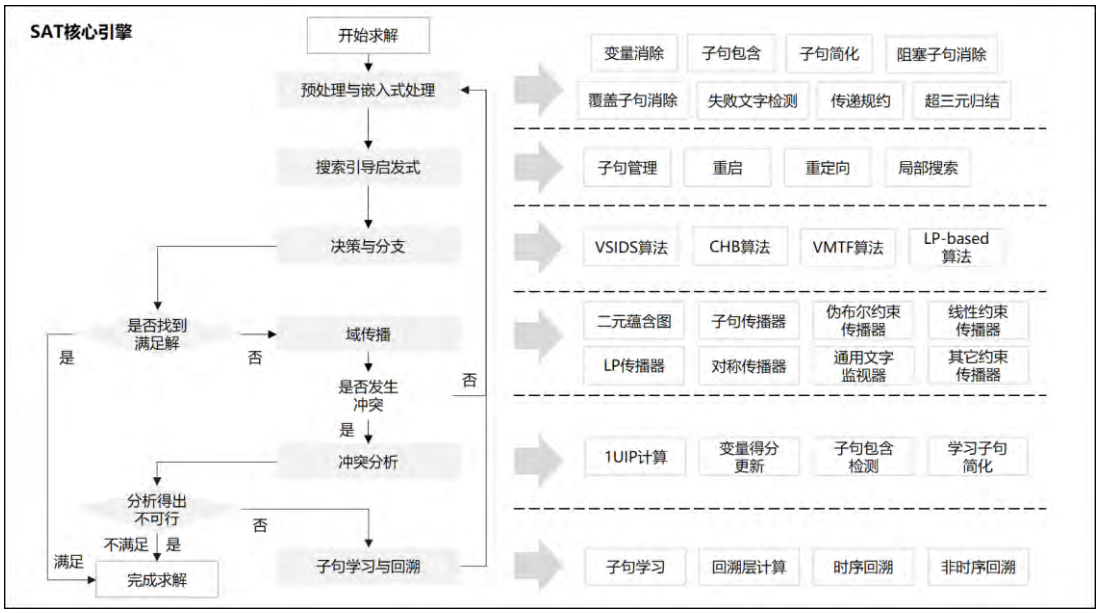

图2.14 SAT核心引擎算法流程

3) 决策与分支。决定下一个分支变量及其赋值。经典的算法包括VSIDS（Variable State Independent Decaying Sum）[53]、CHB（Conflict History-Based branching heuristic）[54]、VMTF（Variable Move To Front）[55]等来自于SAT求解器的启发式方法，以及来自于数学规划求解器的参考松弛解的LP-based算法；

4) 域传播。基于各类约束和已有的变量赋值，推导出新的变量赋值或使得变量的域缩紧，当变量域为空时即得到冲突。在约束规划求解器中，存在着许多的约束类型，针对不同类型的约束，有着不同的传播器用于该类约束的传播。例如，针对二元子句的二元蕴含图、针对非二元子句的子句传播器、针对伪布尔约束和线性约束的传播器、利用线性规划的LP传播器、利用对称性检测的对称传播器、通用文字传播器及其它各类约束传播器等；

5) 冲突分析。分析冲突产生的原因，归纳出新的学习子句。冲突分析通常采用基于首个唯一蕴含点（1UIP）的技术对冲突蕴含图进行分析，得出一条学习子句。在这一过程中涉及到的变量得分会被更新，使得后续的分支中这些变量被选择的概率增加。此外，在冲突分析过程中，还设计学习子句包含的检测，以及学习子句的化简等技术；

6) 子句学习与回溯。将学习子句加入公式中，并回溯到相应层继续搜索。首先

将冲突分析得到的学习子句加入公式，以避免相同冲突再次出现。随后进行回溯层计算，确定应回退到的决策层，通常是学习子句中次高层次的变量所在层。接着执行回溯操作，弹出赋值轨迹中高于该层的赋值，并在目标层强制传播学习子句。回溯方式分为两类：时序回溯仅逐层回退，而非时序回溯可直接跳到合适层次，更高效地剪枝搜索空间。

（四）启发式算法

启发式算法是不依赖完整系统化搜索的求解策略，而是通过对当前解进行连续改进来逼近高质量可行解的技术，其中最典型的就是局部搜索和大邻域搜索，除此之外还有基于变量固定与传播的启发式算法，以及精确算法变种启发式，即对精确算法进行参数调整得到的快速启发式算法。值得注意的是，大邻域搜索可被视为一种特殊的固定与传播启发式，两者可共用一套流程与框架。启发式算法的核心优势在于，它们不追求穷举搜索，而是依靠结构化的邻域设计、随机性和增量评估机制，在巨大搜索空间中高效探索，因此特别适合求解规模大、约束复杂、对最优性要求不严格的实际应用场景。局部搜索与大邻域搜索/固定与传播启发式算法流程图如图2.15所示。

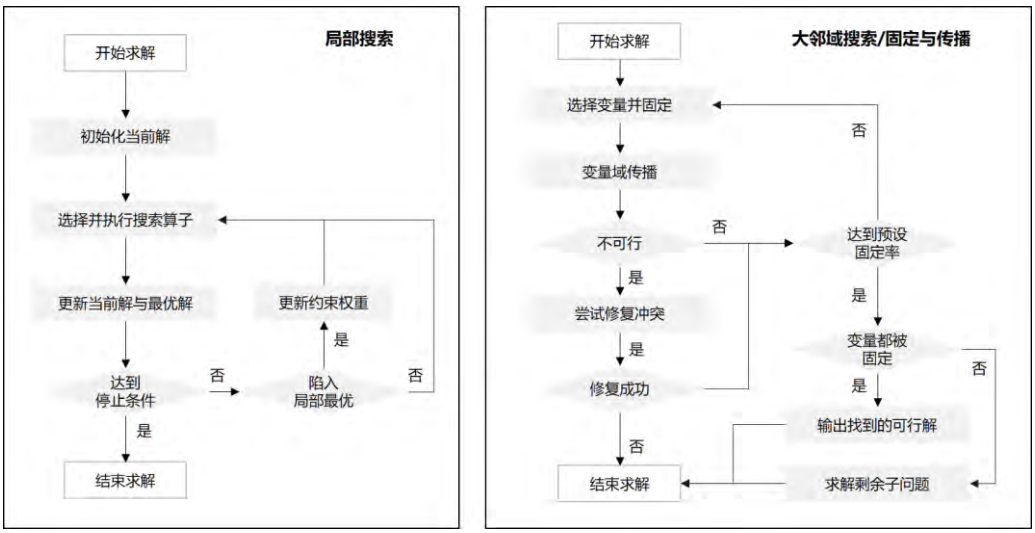

图2.15 约束规划启发式算法流程

1) 局部搜索。从一个初始解出发，通过在其邻域内进行小幅度修改，如交换两个任务、调整一个变量的值，来寻找更好的解，当遇到局部最优时则通过更新约束权重或随机扰动等策略跳出困境。不同的初始解生成、搜索算子、逃离局部最优的策略可衍生出不同的局部搜索算法。此外，针对不同类型问题也可设计与采用不同

的局部搜索算法进行求解。通常来说，特定问题算法的性能远好于通用算法的性能；

2) 大邻域搜索/固定与传播。每轮迭代选择一个变量进行固定，而后进行域传播操作，若得到不可行，则尝试修复当前冲突，若修复成功或传播未得到不可行均判断是否达到算法预设的固定率，若达到了固定率，当变量全部都被固定时，该算法相当于一个固定与传播启发式算法，只需输出找到的可行解即可；否则相当于一个大邻域搜索启发式算法，此时算法需要求解剩余未固定变量所组成的子问题。固定与传播算法利用了约束规划求解器的强推理能力，而大邻域搜索结合了局部搜索的灵活性与约束规划的强推理能力，能够在复杂调度、路径规划、资源分配等问题中快速找到高质量解。

（五）数学规划技术融合

在现代优化与规划领域，基于CP-SAT技术的混合约束规划求解器代表了一种跨学科融合的前沿方法，它不仅继承了计算机科学中的形式化建模与算法设计思想，还吸收了运筹学、人工智能以及逻辑推理等多个领域的成果，从而能够在复杂约束环境下高效地求解大规模问题。其核心优势在于将不同类型的求解技术有机结合，形成一个能够处理复杂约束耦合问题的统一框架。

数学规划、约束满足系列问题从特殊到一般分别是SAT（可满足性问题）、MaxSAT（最大可满足性问题）、PBO（伪布尔优化）、ILP（整数线性规划）、MILP（混合整数线性规划）/CIP（约束整数规划）、CP（约束规划），可以看出SAT和MaxSAT就是ILP、MILP、CIP、CP等一系列通用优化问题的核心基础。下图所示为CP-SAT的技术融合示意图，可见基于CP-SAT的混合约束规划求解器融合了来自SAT、CP、MILP等领域的关键技术，各领域的核心作用如图2.16所示。

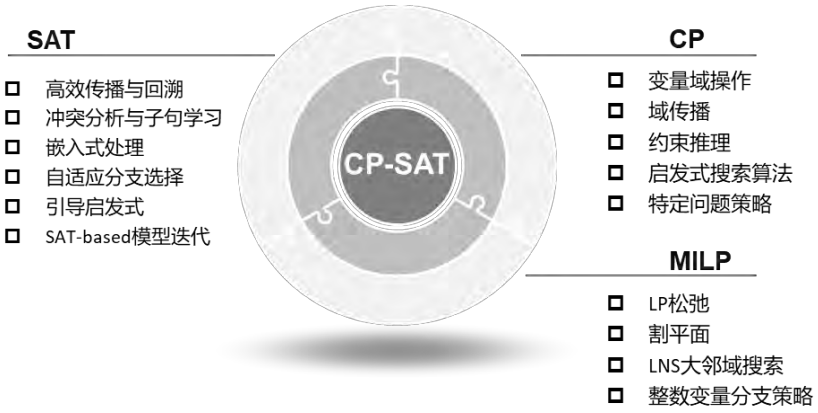

图2.16 CP-SAT求解器技术融合示意图

1) 可满足性问题SAT。SAT技术为混合求解器提供了逻辑层面的强大支撑。通过布尔变量和子句的形式化表达，问题被转化为逻辑可满足性判定。现代SAT求解器依赖于冲突驱动子句学习框架，其中包括单元传播、冲突分析、学习子句生成等关键机制。懒惰子句生成框架可将各种复杂约束耦合在一起，在传播遇到冲突时根据需要生成学习子句。由于SAT求解技术的高效性与先进性，传统约束规划传播与搜索效率低的问题得到显著改善；

2) 约束规划CP。CP强调通过约束传播与搜索策略来缩小解空间。其核心思想是利用全局约束和传播机制，在搜索过程中不断剪枝，避免无效分支的扩展。CP的优势在于灵活的建模能力，能够自然表达复杂的离散约束关系。通过传播算法，求解器可以在变量域上快速排除不可能的取值，从而显著减少搜索空间。此外，CP常结合启发式搜索策略，使得求解过程更具针对性和高效性。对于排班、调度、装箱等结构化问题，CP的全局约束和传播机制往往能发挥极大作用；

3) 混合整数线性规划MILP。MILP模块融合了线性规划与整数规划的优势，成为处理数值约束与组合优化的核心工具。在混合求解器中，MILP的作用不仅在于直接求解整数线性模型，更重要的是通过 LP 松弛解提供下界，从而在搜索过程中实现有效剪枝。基于松弛解的下界计算，使得求解器能够快速判断某些分支是否值得继续探索。与此同时，MILP技术还包括割平面方法：用于不断强化模型，缩小可行域；大领域搜索启发式：求解子问题的启发式算法可帮助求解器快速得到高质量可行解；整数变量分支策略：基于LP解为搜索树的分支提供更多策略。通过这些技术，MILP模块在混合求解器中承担了提供数值下界、强化约束以及全局优化的关键角色。

（六）并行

并行与分布式算法旨在利用多核或多机资源加速求解，并行与分布式运行架构如图2.17所示，同步与调度器负责从多个线程或机器同步信息，并重新进行任务调度，各线程或机器执行各自的任务，并在同步点等待同步与下一步指令信息。

图2.17 约束规划求解器并行架构

并行与分布式的核心思想主要体现在Portfolio与Divide and Conquer（分而治之）两大策略中：

1) Portfolio策略。Portfolio的理念是多样性即力量，同时运行多个具有不同启发式、不同搜索策略或不同参数配置的求解器实例，让它们在同一问题上并行竞争，谁先找到解就终止其他实例。这种方法不需要拆分问题本身，而是依赖策略多样性来覆盖更广的搜索空间，因此对复杂组合问题尤其有效。在上图中，Portfolio策略对应的不同任务就是用不同的方法求解同一个原始问题；

2) Divide and Conquer策略。相比之下，Divide and Conquer方法则直接将原问题的搜索空间划分为多个互不重叠的子空间，并将这些子任务分配给不同的线程或节点独立求解，最终将结果合并，对于可行性问题若任何子空间找到可行解则整体成功；对于优化问题则需比较各子空间的最优值。该方法的关键在于如何高效划分搜索树、避免负载不均以及确保子空间之间无重复搜索。在上图中，Divide and Conquer策略对应的不同任务就是原问题分解得到的不同子问题。

此外，上述两者的结合也能够充分发挥两者的互补性，使约束规划求解器能够充分利用现代计算平台的并行能力，在大规模调度、资源分配和组合优化任务中显著提升求解速度与鲁棒性。

2.1.3 黑箱优化求解器技术

2.1.3.1 黑箱优化求解器简介

黑箱优化求解器（Black-Box Optimization Solver）是一类针对目标函数解析式未知、无梯度信息场景的通用优化工具，核心通过函数评估结果驱动寻优进程，无需依赖数学建模推导，可广泛适配连续、离散、混合类型参数的优化任务，在工业设计、超参数调优、路径规划等复杂场景中具有显著适用性。与传统梯度优化方法不同，黑箱优化求解器更注重对搜索过程的自适应调控与样本效率提升，能够在高维、非平滑、含噪声的复杂优化空间中高效探索可行解与最优解。

黑箱优化求解器的核心寻优流程遵循“迭代闭环、自适应调控”的底层逻辑，整体可划分为三大核心阶段，各阶段协同衔接形成完整优化链路：

第一阶段为初始化建模与采样。首先对优化目标的搜索空间进行结构化建模，支持浮点型、整型、类别型等多类型参数的边界约束与配置定义，而后采用拉丁超立方采样（Latin hypercube sampling，LHS）[56]等高效采样策略生成少量初始样本点。该阶段的核心目标是在控制初始评估成本的前提下，最大化样本对解空间的覆盖度，为后续迭代优化提供高质量初始数据支撑，实现解空间覆盖度与初始评估效率的精准平衡。

第二阶段为迭代闭环寻优。以“采样-评估-更新”的循环模式推进优化进程：由策略调度模块基于历史评估数据生成下一轮候选解，优化执行模块调用目标函数完成候选解性能评估，同步将评估结果反馈至模型模块，用于更新概率分布模型或迭代优化策略，动态调整探索与利用（Exploration-Exploitation）的权重。整个迭代过程无需人工干预，可通过自适应机制适配不同优化场景的特性。

第三阶段为收敛判定与结果输出。基于多维度终止准则自动判定寻优收敛状态，核心准则包括连续迭代函数值变化阈值（如连续n次迭代差值小于ϵ）、最大评估次数、时间预算等，满足任一准则即终止寻优流程。最终输出最优参数配置、目标函数最优值及完整历史评估曲线，同时可通过早停机制、超参数自适应更新等优化手段，在有限评估成本约束下，实现全局最优解的高效探索与精准定位。

2.1.3.2 黑箱优化求解器软件架构

如下图所示，黑箱优化求解器采用模块化架构设计，分为问题定义、优化器、可视化工具三个模块，各模块间采用松耦合设计，具备良好的算法扩展性与跨场景适配能力，可灵活对接不同领域的优化需求。

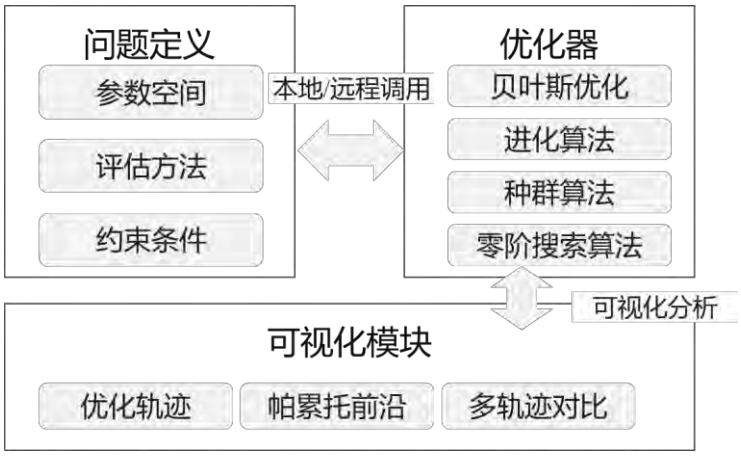

图2.18 黑箱优化求解器软件架构

问题定义模块负责实现优化问题的建模、定义与扩展适配，为后续优化计算提供标准化的问题输入，核心构成包括基础核心类、远程扩展类及辅助工具组件，形成完整的问题定义能力体系。

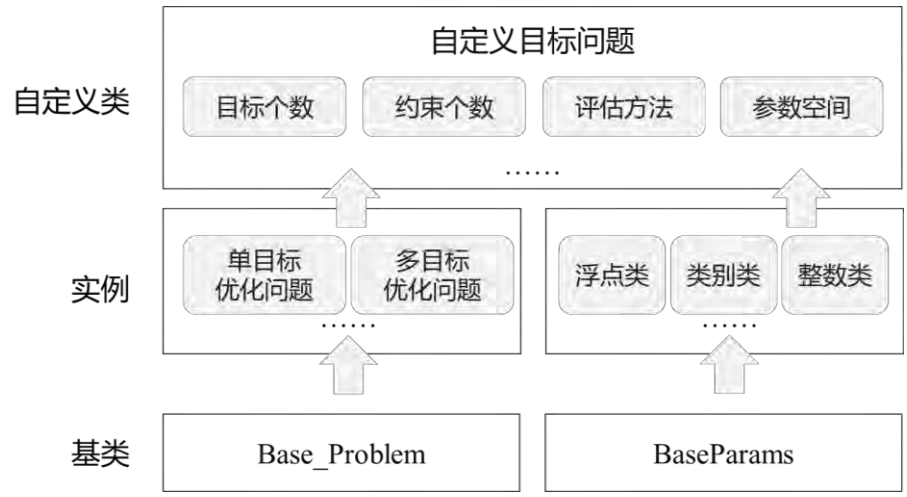

图2.19 问题模块架构

1) 基础核心：以problem.base.Base_Problem为自定义问题基类，提供统一的问题抽象接口，适配各类优化场景，是问题定义的核心支撑；

2) 远程扩展：通过problem.remote.RemoteProblem远程评估问题类，实现远程目标函数调用与评估结果回收，适配分布式部署场景；

3) 辅助工具：集成Params参数配置接口与内置测试目标函数库，支持参数规范化配置与算法性能快速验证。

优化器模块作为求解器的核心计算中枢，封装各类优化算法及辅助组件，负责实现从候选解生成、迭代优化到结果输出的全流程计算，是保障寻优精度与效率的核心单元。

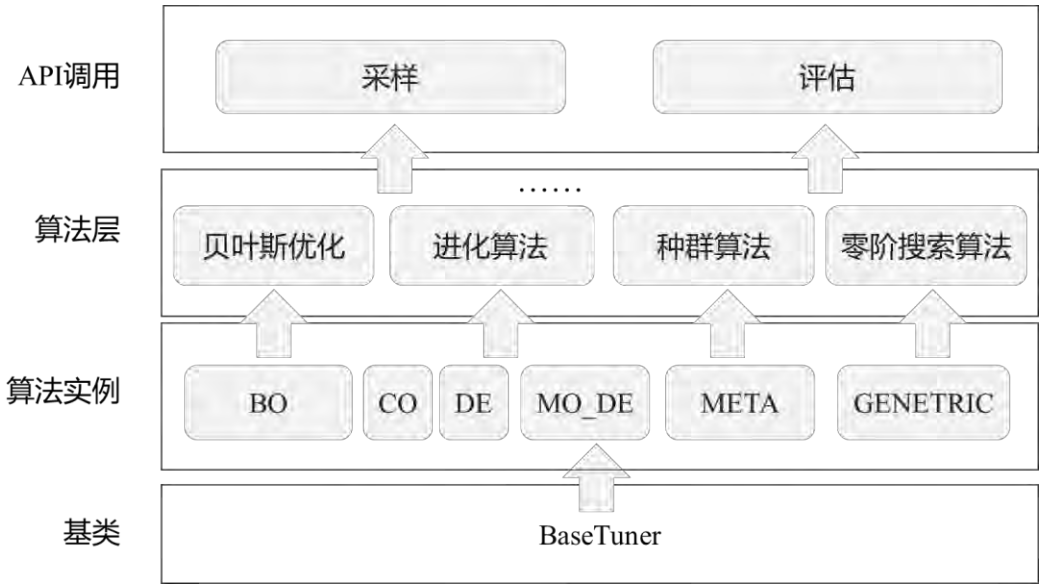

图2.20 优化器模块架构

1) 核心优化器：以BayesianOptimizationTuner（贝叶斯优化器）为核心，具备自适应寻优能力，适配复杂优化场景并支持算法灵活切换；

2) 子模块集合：集成BO（Bayesian Optimization）、CO（Collaborative optimization）、DE（Differential Evolution）、MO_DE（Multi-Objective Differential Evolution）等多类算法子模块，可根据问题类型匹配适配的子模块，形成多算法协同优化能力；

3) 辅助组件：配套采集函数库与工具库，平衡寻优探索与利用，保障优化过程稳定可追溯。

可视化模块聚焦优化过程与结果的可视化呈现，提供标准化的可视化接口，助力用户解读寻优逻辑、验证算法性能，核心由可视化主类与功能接口构成。

1) 核心接口：以visualization.Visualizer为可视化主类，提供统一入口，支持与其他模块数据联动，简化可视化流程；

2) 功能接口：封装日志加载、收敛曲线绘制、帕累托前沿可视化等专用接口，

覆盖单目标、多目标优化可视化需求，支持结果导出。

三大模块通过标准化接口协同工作，形成“问题定义-优化计算-可视化分析”的闭环链路，适配多场景需求，实现全流程覆盖。

2.1.3.3 黑箱优化求解器求解流程

黑箱优化求解器的整体求解流程遵循目标定义、参数配置、启动初始化、迭代寻优的核心逻辑，整体求解流程如下图所示：

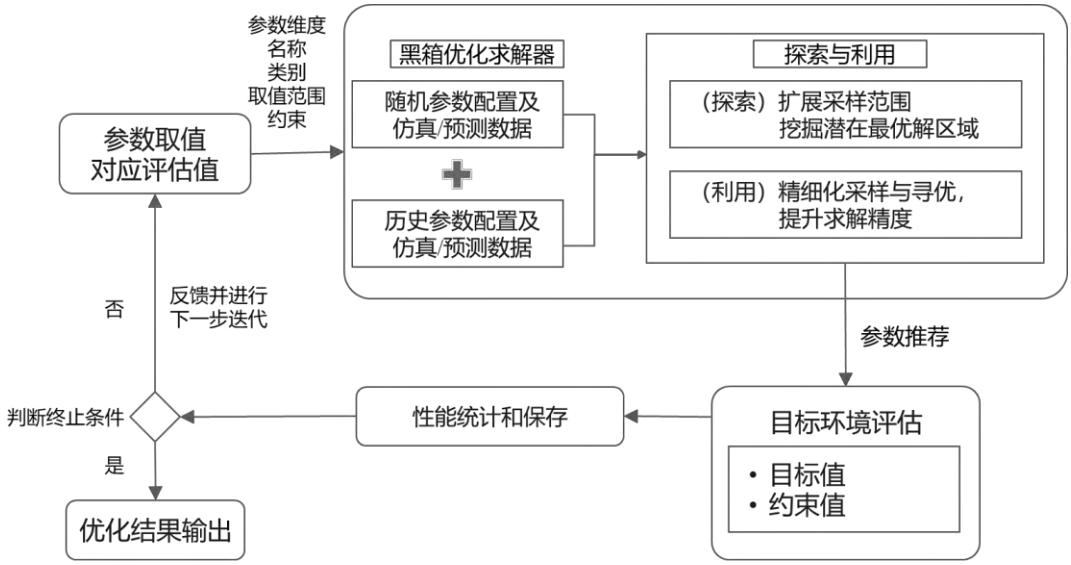

图2.21 黑箱优化求解流程

第一步为目标问题定义，该阶段核心是完成优化问题的规范化建模，具体包括优化变量的维度、定义域及类型定义，目标函数的评估接口与性能指标定义，约束条件（如果有的话）的数学描述与满足判定准则定义，同时明确目标问题的优化方向（最小化/最大化），为后续求解过程提供明确的问题边界与评估标准。

第二步为优化器设置，基于目标问题的特性（如变量维度、是否含约束、优化精度要求等）完成优化策略选型，同时进行各功能模块与超参数的精细化配置。例如，若选用贝叶斯优化策略，需完成代理模型选型、采集函数配置、采集函数优化器选型；若选用进化类优化策略，需配置种群规模、交叉概率、变异概率等核心超参数，确保优化器适配目标问题的求解需求。

第三步为启动优化过程，由于黑箱优化问题中目标函数的解析表达式未知、无先验知识可依托，需通过初始化采样完成求解启动。具体可采用随机均匀采样生成

初始样本集，或基于历史同类问题数据进行迁移学习，通过预训练代理模型实现初始采样的高效引导，该阶段需投入一定的探索成本与数据采集成本，核心目标是获取初始有效样本，为后续迭代优化提供基础数据支撑。

第四步为迭代优化，该阶段是黑箱优化求解的核心环节。基于启动阶段获取的初始样本，优化器交替执行探索与利用：探索通过扩展采样范围、更新代理模型（若采用代理模型类策略）挖掘潜在最优解区域，利用在当前最优解邻域内进行精细化采样与寻优，提升求解精度。迭代过程中，优化器会定期进行策略交互，更新样本集、代理模型参数、当前最优解及目标函数上下界，同时实时判定是否满足终止条件。若当前最优解达到预设精度、连续多轮迭代目标函数提升量低于阈值，或达到最大采样次数、最长计算时间限制，或判定问题无可行解，则迭代终止，输出最优解；否则继续迭代，直至满足终止条件。

2.1.3.4 黑箱优化求解器功能特性

黑箱优化求解器作为用户求解黑箱优化问题的核心工具，依托其分层模块化架构，提供从问题建模、迭代寻优到结果分析的全流程功能，既覆盖基础操作需求，也具备高阶扩展能力，适配学术验证、工业应用等多场景，具体功能如下：

1) 问题建模支持：支持连续、离散、混合类型参数的优化问题建模，兼容带约束/无约束、单目标/多目标等各类任务场景；提供自定义目标函数封装接口，同时内置Ackley[57]、ZDT1[58]等经典测试函数，可直接用于算法性能验证，无需额外搭建测试环境；

2) 远程与扩展能力：支持远程目标函数的调用与评估结果回收，适配分布式部署场景，解决本地资源1不足、目标函数无法本地运行等痛点；各模块松耦合设计，支持核心功能单独调用与二次开发，可灵活对接不同领域优化需求；

3) 优化求解能力：以贝叶斯优化器为核心，集成差分进化等多类优化算法，可根据问题类型自动匹配或手动选择适配算法，具备自适应寻优能力，适配高维、非平滑、含噪声等复杂场景；支持约束优化处理，保障寻优精度与效率；

4) 参数与过程管理：支持搜索空间边界、评估预算、终止准则等参数的规范化配置与解析；配备日志打印、异常捕获、参数校验等工具，可实时监控优化过程，保障优化过程稳定可追溯，支持优化任务中断并返回当前结果；

5) 可视化与结果输出：提供统一的可视化入口，支持优化日志加载、收敛曲线绘制、帕累托前沿可视化等功能，直观呈现优化过程与结果；支持优化结果导出与二次分析，助力用户快速解读寻优逻辑、验证算法性能。

2.1.3.5 贝叶斯优化关键技术

贝叶斯优化是一种基于贝叶斯定理的全局优化方法，专门用于解决目标函数计算昂贵、无显式解析表达式、梯度不可得的黑箱优化问题。它通过构建概率代理模型（通常为高斯过程）[59]对目标函数进行建模，利用采集函数指导后续采样方向，在探索未开发区域与利用已知最优区域之间进行平衡，以最少的评估次数逼近全局最优解。其核心目标是在有限的评估预算内，找到使目标函数$f(x)$尽可能接近全局最优值f^*的近似最优解x_{best}。

贝叶斯优化技术的基本特点包括：

1) 序列优化特性：采用逐次迭代的采样方式，每次采样都基于历史观测数据和代理模型的预测结果，而非随机采样，大幅提升采样效率；

2) 概率建模核心：以高斯过程为核心代理模型，不仅能预测未知点的函数值，还能提供预测不确定性，为采集函数的设计提供支撑；

3) 探索与利用平衡：通过采集函数（如期望改进EI、上置信边界UCB等）动态权衡“探索”（未探索区域的不确定性）和“利用”（已知最优区域的潜力），避免陷入局部最优；

4) 依赖先验知识：可结合目标函数的先验信息（如平滑性、定义域特性）选择合适的核函数，进一步提升模型拟合精度和寻优效率。

利用贝叶斯优化求解黑箱优化问题时，其核心优势一般包含如下几点：

1) 样本高效性：无需大量评估目标函数，仅通过少量采样即可逐步逼近全局最优解，尤其适用于单次评估成本高、样本稀缺的场景，比随机搜索、网格搜索更高效；

2) 全局寻优能力：通过概率模型的不确定性估计，能够有效探索搜索空间，避免陷入局部最优陷阱，适合非凸、多极值的黑箱优化问题；

3) 灵活性强：支持噪声与无噪声场景，可通过稳健高斯过程等模型适配含噪声的工业场景，同时可根据问题特性选择不同的采集函数和核函数；

4) 可解释性较好：概率代理模型能够提供预测不确定性，便于用户判断寻优过程的可靠性和后续采样的潜在价值。

同时，贝叶斯优化也存在一定局限性：计算复杂度较高（高斯过程逆矩阵计算为$O(n^3)$），对高维问题扩展性有限，对采集函数的选择较为敏感，且假设目标函数具有平滑性，在高度不规则的函数上效果可能不佳。

贝叶斯优化的算法流程一般为如下步骤：

1) 初始化

设定总评估预算N，选择合适的采集函数（如期望改进EI、上置信边界UCB等）及核函数（如平方指数核、Matérn核等）311[59]，核函数用于刻画样本间的相似性。

通过拉丁超立方采样等初始设计方法，在搜索空间$\mathcal{X}$内选取n_{init}个初始点，构成初始数据集$D_{1:n_{\text{init}}} = \{(x^{(i)}, y^{(i)})\}_{i=1,\dots,n_{\text{init}}}$，其中$y^{(i)} = f(x^{(i)}) + \epsilon$（$\epsilon$为噪声项，无噪声场景下$\epsilon = 0$）。

令当前迭代次数$t = n_{\text{init}}$，完成初始评估。

2) 主循环（当$t < N$时）

模型更新：基于当前数据集$D_{1:t}$，训练或更新高斯过程代理模型P_t，其中$P_t(f) \sim GP(\mu_t(x), k_t(x, x'))$，$\mu_t(x)$为预测均值函数，$k_t(x, x')$为协方差函数。

采集函数优化：通过梯度法或启发式算法优化采集函数，确定下一个评估点$x^{(t+1)} = \text{argmax}_{x \in \mathcal{X}}\, a_t(x)$，其中$a_t(x)$由$P_t$的预测均值$\mu_t(x)$和不确定性$\sigma_t(x)$共同决定，实现探索与利用的平衡。

黑盒评估与数据更新：评估$x^{(t+1)}$处的真实目标函数值$y^{(t+1)} = f(x^{(t+1)}) + \epsilon$，将新观测对$(x^{(t+1)}, y^{(t+1)})$加入数据集，更新为$D_{1:t+1} = D_{1:t} \cup \{x^{(t+1)}, y^{(t+1)}\}$，并更新$t = t + 1$。

3) 输出

当评估次数t达到总评估预算N时，终止迭代，输出整个优化过程中的历史最优解：$x_{\text{best}} = \text{argmin}_{x \in \{x^{(1)}, \dots, x^{(N)}\}} f(x)$。

在更多的实际复杂场景中应用时，贝叶斯优化也可以相应地设计若干高阶特性用于更好地支持复杂黑箱优化场景的求解，包括多策略探索、鲁棒优化、流形优

化、多阶段优化等。

4) 多策略探索

多策略探索是贝叶斯优化的高阶特性之一，核心是通过配置多个代理模型、采集函数或采集函数优化器，解决单一策略通用性不足的问题，提升优化的稳定性和效率，尤其适用于高维复杂搜索空间中全局搜索与局部精调的平衡场景。

MACE（Multi-objective Acquisition function Ensemble）是典型的多策略探索方法[60]，该方法将EI、PI、UCB等常用采集函数进行聚合，在三种采集函数的帕累托前沿上进行采样，既避免了单一采集函数的局限性，还支持并行探索。在深度学习超参数优化等场景中，可结合随机采样、梯度引导和贝叶斯模型预测进行动态调整，大幅提升寻优效率。

5) 鲁棒优化

鲁棒优化主要用于应对数据噪声、对抗扰动等实际场景中的不确定性问题（如自动驾驶中传感器故障导致的状态观测误差），核心是通过变分贝叶斯推断抑制损坏数据的影响，在一系列代理函数分布中寻找表现稳定的候选点，确保优化结果的可靠性。

鲁棒优化的核心逻辑的是：针对受噪声影响的目标问题，通过调整贝叶斯优化器配置，降低噪声对寻优过程的干扰。例如，当目标函数存在随距离某个稳定点变化的噪声时，鲁棒优化可在逼近稳定点的同时，最小化期望反馈误差，实现稳定寻优。

6) 流形优化

流形优化适用于具有内在低维结构的高维黑箱优化问题，核心思想是学习高维数据背后的低维潜在流形结构，将原本复杂的高维优化问题转化为低维流形上的优化问题，从而减少昂贵的目标函数评估次数，提升寻优效率和质量。

一类典型的流形优化的核心实现流程如下：

热启动数据采集：先通过随机采样获取一定数量的合法候选点，作为流形学习的训练数据。

流形模型训练：通过训练VAE（变分自编码器）模型[61]，将高维输入空间映射到规则的低维表征空间，实现高维到低维的转化。

低维寻优与映射：在训练好的低维流形上进行贝叶斯优化，得到低维最优解后，

通过VAE解码器映射回原高维空间，得到最终的优化结果。

训练好的VAE模型可以在相同问题场景中复用，适用于高维约束优化、复杂系统参数优化等场景。

7) 多阶段优化

多阶段优化通过分阶段、分层次的寻优策略，平衡优化精度与评估成本，适用于目标函数评估代价高、存在多源数据或多保真度场景的黑箱优化问题，核心分为预训练代理模型和多保真度优化两个关键阶段。

预训练代理模型用于实现贝叶斯优化的热启动，适用于交互代价较高的环境或仿真器。核心是选取支持增量训练的深度网络代理模型，通过历史数据（或随机生成的同构数据）预训练代理模型，减少寻优初期的样本采集成本，加速参数搜索过程。

多保真度优化则针对“完整评估代价高、近似评估代价低”的场景，通过融合不同保真度的目标问题数据，在保证优化精度的同时降低评估成本。核心是引入高保真度（评估准确但成本高）和低保真度（评估近似但成本低）两类目标问题，通过配置参数控制两类数据的使用比例，实现多源数据融合寻优，适用于高成本评估的工业仿真、工程设计等场景。

2.1.3.6 进化算法关键技术

进化算法是一大类受生物进化论“自然选择、适者生存”思想启发的随机搜索算法，核心是通过模拟生物种群的进化过程，对“个体”（即优化问题的解）执行“选择、交叉、变异”等遗传操作，使种群一代代地适应优化目标，逐步逼近全局最优解。它不依赖目标函数的梯度信息，属于无梯度优化方法，适用于复杂、多极值、高维的黑箱优化问题。其基本特点包括：

1) 种群搜索机制：以“种群”为搜索单元，而非单个解，种群内个体的多样性确保了搜索空间的全面性，降低陷入局部最优的概率；

2) 遗传操作核心：通过选择、交叉、变异三大核心操作实现种群更新，选择保留优秀个体，交叉实现基因重组，变异引入新的基因信息，维持种群多样性；

3) 随机性与鲁棒性：搜索过程具有随机性，能够适应复杂多变的搜索空间，对目标函数的连续性、可导性无要求，鲁棒性强；

4) 迭代进化特性：通过多代迭代逐步优化种群适应度，无需提前掌握目标函数的内在规律，仅通过适应度（目标函数值）判断个体优劣。

进化算法的核心优势包括：

1) 全局搜索能力强：种群多样性和随机搜索特性使其能够全面探索搜索空间，有效避免陷入局部最优，尤其适用于多极值、非凸的复杂黑箱问题；

2) 无梯度依赖：无需计算目标函数的梯度，可处理无显式表达式、不可导的黑箱优化问题，适用范围极广；

3) 鲁棒性突出：对目标函数的噪声、非线性、离散性不敏感，能够适配工业制造、工程设计等复杂场景中的不确定性问题；

4) 易于并行实现：种群内个体的评估的可独立进行，可结合并行优化技术，充分利用硬件资源，提升寻优效率；

5) 灵活性高：可根据问题特性调整遗传操作的参数（如交叉概率、变异概率），或融合其他算法思想，形成混合优化策略。

进化算法的一般流程包括以下步骤：

1) 初始化

设定总评估预算N、种群大小M、交叉概率p_c、变异概率p_m等关键参数，其中交叉概率控制交叉操作的频率，变异概率控制变异操作的频率。

在搜索空间$\mathcal{X}$内随机生成初始种群$P^{(1)} = \left\{x_1^{(1)}, \dots, x_M^{(1)}\right\}$，确保初始种群在搜索空间内均匀分布，保障搜索的全面性。

评估初始种群中每个个体的适应度（即目标函数值）：$y_i^{(1)} = f\left(x_i^{(1)}\right)$，$i = 1, \dots, M$。令当前代数$g = 1$，已用评估次数$t = M$。

2) 主循环（当$t < N$时）

选择（父代选择）：根据个体的适应度$y_i^{(g)}$，以较高概率选择优秀个体进入交配池$S^{(g)}$，常用轮盘赌选择、锦标赛选择等方法[62]，确保优秀基因得以遗传。

交叉（重组）：对交配池$S^{(g)}$中的个体两两配对，以概率p_c交换部分基因信息，生成临时子代种群$C^{(g)}$，常用单点交叉、模拟二进制交叉[63]等方法，增加种群多样性。

变异：对临时子代种群$C^{(g)}$中的每个个体，以概率p_m对其基因进行随机扰动，生成最终子代种群$O^{(g)}$，常用高斯变异、多项式变异等方法[64]，避免种群陷入局部最优。

评估与环境选择：评估子代种群$O^{(g)}$中每个个体的适应度y，设定已用评估次数$t = t + \left|O^{(g)}\right|$。从父代$P^{(g)}$和/或子代$O^{(g)}$中选出M个适应度最优的个体，构成新一代种群$P^{(g+1)}$，更新代数$g = g + 1$。

3) 输出

当评估次数达到总预算N时，终止迭代，输出整个进化过程中适应度最好的个体：$x_{\text{best}} = \text{argmin}_{x \in \cup_g P^{(g)}} f(x)$。

2.1.3.7 种群算法关键技术

种群算法（又称种群智能算法）是一类模拟生物群体（如鸟群、鱼群、蚁群）集体行为的随机优化算法，核心是通过个体间的简单交互和社会信息共享，引导整个群体向最优区域移动。与经典进化算法不同，其个体更新不依赖“选择、交叉、变异”等遗传操作，而是基于群体内的信息传递与协同，具有收敛速度快、鲁棒性强的特点，适用于大规模、多约束的黑箱优化问题。标准粒子群优化（PSO）是种群算法的典型代表[1]。其基本特点包括：

1) 群体协同性：个体通过共享自身的历史最优信息和群体的全局最优信息，实现协同寻优，无需复杂的遗传操作，交互逻辑简单；

2) 社会信息驱动：个体的位置更新依赖“个体历史最优”和“群体全局最优”两个核心信息，体现了生物群体的“从众性”和“学习性”；

3) 收敛速度快：相比进化算法，无需经过多代遗传迭代，个体可快速向最优区域靠拢，尤其适用于大规模优化问题；

4) 参数易调节：核心参数（如惯性权重、学习因子）物理意义明确，调节简单，可根据问题特性快速适配，易于工程实现。

种群算法的核心优势包括：

1) 收敛速度快：个体直接向最优区域移动，无需复杂的遗传操作，迭代效率

高，可快速逼近最优解，适合大规模优化场景；

2) 鲁棒性强：对搜索空间的拓扑结构、目标函数的非线性、噪声不敏感，能够适应复杂多变的优化场景，稳定性好；

3) 易于实现：算法逻辑简单，无需设计复杂的遗传操作，代码实现难度低，工程落地成本低；

4) 并行性好：群体内个体的位置更新和适应度评估可独立进行，可结合并行优化技术，进一步提升寻优效率；

5) 多约束适配性强：可通过简单的约束处理机制，适配大规模、多约束的黑箱优化问题，无需复杂的约束转化。

下面以标准粒子群优化（PSO）法为例，介绍种群算法的算法流程：

1) 初始化

设定总评估预算N、种群大小M、惯性权重w、学习因子c_1和c_2等关键参数，其中惯性权重控制粒子的惯性运动，学习因子控制粒子向个体最优和全局最优的学习强度。

对于每个粒子$i(i=1,\dots,M)$：（1）随机初始化其位置$x_i^{(1)} \in \mathcal{X}$和速度$v_i^{(1)}$（常设为0或小随机值）；（2）评估其初始适应度：$y_i^{(1)} = f\left(x_i^{(1)}\right)$；（3）初始化其个体历史最优位置和最优值：$p_{\text{best},i} = x_i^{(1)}$，$g_{\text{best}_{\text{value}},i} = y_i^{(1)}$。

确定整个种群的全局历史最优位置和最优值：$g_{\text{best}} = \text{argmin}_{p_{\text{best},i}}\left(p_{\text{best\_value},i}\right)$，$g_{\text{best\_value}} = \min_i\left(p_{\text{best\_value},i}\right)$。令当前迭代次数t=M。

2) 主循环（当$t < N$时）

粒子状态更新：对于每个粒子i，更新其速度与位置，更新公式为：$v_i^{(t+1)} = w \cdot v_i^{(t)} + c_1 r_1 \cdot \left(p_{\text{best},i} - x_i^{(t)}\right) + c_2 r_2 \cdot \left(g_{\text{best}} - x_i^{(t)}\right)$，$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}$，其中$r_1, r_2 \sim U(0,1)$为随机数，增加搜索的随机性。

粒子评估：评估每个粒子新位置$x_i^{(t+1)}$的适应度：$y_i^{(t+1)} = f(x_i^{(t+1)})$，更新已用评估次数$t = t + 1$。

最优信息更新：对每个粒子i，若$y_i^{(t+1)} < p_{\text{best_value},i}$，则更新个体历史最优：

$p_{\text{best},i} = x_i^{(t+1)}$,$p_{\text{best\_value},i} = y_i^{(t+1)}$；检查并更新全局最优：若$\min_i(p_{\text{best_value},i}) < g_{\text{best_value}}$，则更新$g_{\text{best}}$和$g_{\text{best_value}}$。

3) 输出

当评估次数t达到总评估预算N时，终止迭代，输出全局历史最优位置：$x_{\text{best}} = g_{\text{best}}$。

2.1.3.8 零阶搜索算法关键技术

零阶搜索算法是一类不依赖目标函数梯度信息，仅通过直接比较函数值来指导搜索方向的无梯度优化方法，核心是通过在当前最优解周围进行采样探测，根据采样点的函数值好坏调整搜索方向和步长，逐步逼近最优解。其算法结构简单、易于实现，适用于低维、评估成本较低的黑箱优化问题，模式搜索是零阶搜索算法的典型代表[65]。其算法基本特征包括：

1) 无梯度依赖：无需计算目标函数的梯度，仅通过比较采样点的函数值即可确定搜索方向，适用于不可导、无显式表达式的黑箱问题；

2) 局部探测为主：搜索过程围绕当前最优解展开，通过局部采样探测更优点，搜索逻辑简单，计算成本低；

3) 步长自适应调整：通过步长收缩、扩展系数，根据探测结果动态调整搜索步长，平衡搜索精度与速度，成功迭代时扩大步长加速收敛，失败迭代时收缩步长提升精度；

4) 确定性与随机性结合：采样模式（如坐标方向）具有确定性，同时可通过随机采样增加搜索的灵活性，缓解局部最优问题；

5) 局部性特征明显：由于围绕当前点进行探测，具有一定的局部性，可在一定程度上缓解某些优化问题中的多样性不足问题。

零阶搜索算法的核心优势包括：

1) 算法简单易实现：无需复杂的模型构建或遗传操作，仅需设计采样模式和步长调整规则，代码实现难度低，工程落地成本低；

2) 计算成本低：采样逻辑简单，无需大量计算资源，适用于评估成本低、低维的黑箱优化问题；

3) 鲁棒性较好：对目标函数的平滑性、连续性要求低，可处理离散、非凸的黑箱问题，适配简单的工业控制场景；

4) 精度可控：通过步长收缩系数，可灵活调整搜索精度，满足不同场景的优化需求，在局部最优区域可实现高精度寻优；

5) 兼具稳健性：与一阶方法相比，零阶方法具有更强的稳健性，可适配部分含噪声的优化场景。

以模式搜索为例，零阶优化算法的典型流程为：

1) 初始化

设定总评估预算N、初始点$x^{(1)}$、初始步长向量$\delta^{(1)} > 0$、步长收缩系数$\beta \in (0,1)$、步长扩展系数$\gamma > 1$，其中步长系数控制搜索精度与速度。

评估初始点$y^{(1)} = f\left(x^{(1)}\right)$。

设定当前最优点和最优值：$x_{\text{best}} = x^{(1)}$，$y_{\text{best}} = y^{(1)}$，令当前迭代次数$t = 1$。

2) 主循环（当$t < N$时）

探测移动：沿着预定的模式（通常是各坐标轴的正负方向）进行探测，对于每个坐标方向d_k（如标准基向量e_k和-e_k）：（1）生成试探点：$x_{\text{trial}} = x_{\text{best}} + \delta_k^{(t)} \cdot d_k$；（2）评估试探点：$y_{\text{trial}} = f(x_{\text{trial}})$，$t = t + 1$。若$t \geq N$，跳出循环；（3）若$y_{\text{trial}} < y_{\text{best}}$，则接受此次移动：更新$x_{\text{best}} = x_{\text{trial}}$，$y_{\text{best}} = y_{\text{trial}}$，并记录此次成功移动，可立即沿该方向进行模式移动（扩大步长）或继续探测其他方向。

判断与步长更新：（1）成功迭代：如果在本次循环的探测中找到了更优点（即y_{best}被更新），保持或扩大搜索步长，以加速收敛：$\delta^{(t+1)} = \gamma \cdot \delta^{(t)}$，将当前最优点$x_{\text{best}}$作为下一次迭代的基点；（2）失败迭代：如果在本次循环的所有探测中均未找到更优点，则收缩搜索步长，以提高局部精度：$\delta^{(t+1)} = \beta \cdot \delta^{(t)}$，基点$x_{\text{best}}$保持不变。

3) 输出

当评估次数t达到总评估预算N时，终止迭代，输出搜索过程中找到的最优点：x_{best}。

2.2 数值计算求解技术

数值计算求解技术是现代工程与科学计算中不可或缺的一部分，其核心在于通过数值方法求解各种数学模型，包括线性方程组、非线性方程组和特征值问题。这些技术在计算机辅助工程（CAE）、物理仿真、优化设计、控制系统设计、信号处理等领域有着广泛的应用。随着计算能力的不断提升和算法的不断优化，数值计算求解技术在处理大规模、复杂问题方面的能力日益增强，成为推动科学研究和工程实践的重要工具。本节将分别介绍线性方程组求解器技术、非线性方程组求解器技术和特征值求解器技术，探讨它们的基本原理、应用场景和技术路线。这些技术不仅在理论研究中具有重要意义，也在实际工程中发挥着关键作用。

2.2.1 线性方程组求解器技术

2.2.1.1 线性方程组求解简介

线性直接法（Linear Direct Method，LDM）是一类基于矩阵分解求解线性方程组的经典方法，其理论基础可追溯至高斯消元。高斯消元通过对系数矩阵与右端项构成的增广矩阵施加初等行变换，将原始问题转化为等价的三角线性方程组，从而得到方程组的解。

尽管高斯消元在理论上具有明确的计算过程和良好的数值稳定性，但在实际数值计算中通常并不直接采用其原始形式。工程应用中常见的情形是：对同一个系数矩阵 A，需要反复求解多个不同的右端项 b。若对每个右端项均执行一次完整的高斯消元，其计算代价将难以接受。因此，实际应用中通常采用矩阵分解的方式，将稀疏矩阵A分解为若干个简单矩阵的乘积（如三角矩阵、对角矩阵及排列矩阵）的乘积。矩阵分解可以理解为将高斯消元过程“编码”进若干因子矩阵中：对同一系数矩阵，仅需执行一次分解；对于每个右端项，仅通过前向与回代即可获得解，从而显著降低总体计算成本。

在计算机辅助工程（Computer Aided Engineering， CAE）等工程应用场景下，线性方程组的系数矩阵呈现大规模与稀疏性双重特点：大规模体现为矩阵维数 n 量级极高，可达到百万、千万乃至更高；稀疏性则表现为矩阵内绝大多数元素为零，

仅少量非零元素呈非均匀分布。

对于此类问题，传统稠密LU分解的时间复杂度[66]为 $O(n^3)$，空间复杂度为 $O(n^2)$，在时间与内存开销上均不可行。因此，稀疏直接法在分解前通常需要通过排序（Ordering）与符号分析（Symbolic analyze）等预处理手段，尽量降低分解过程中产生的填充（Fill-in），从而将实际计算复杂度控制在远低于稠密分解的水平。进一步地，将具有相同或相似非零结构的若干列（或行）组合成较大的稠密矩阵块，并对这些稠密块执行统一的数值运算，以提高数据局部性与缓存利用率。这种以稠密块为基本计算单元的加速技术通常称为超节点（Supernodal）方法。

与线性直接法不同，迭代法（Iterative Method）并不试图通过显式矩阵分解一次性获得精确解，而是从某一初始猜测 x^0 出发，通过反复修正逐步逼近线性系统的真实解 $x^0 \to x$。

从计算特性上看，迭代法通常以矩阵－向量乘积（Sparse Matrix-Vector Multiplication，SpMV）和向量内积为主要计算核心，其单次迭代的时间复杂度与矩阵非零元个数近似成正比，存储需求也主要由原始矩阵和少量辅助向量构成。这使得迭代法在超大规模稀疏问题中具有明显的内存优势与并行潜力，尤其适合分布式内存和异构计算平台。

然而，迭代法的收敛速度和鲁棒性高度依赖于系数矩阵的谱性质与数值条件。对于病态矩阵或强耦合问题，若缺乏合适的预处理（Precondition），迭代法可能表现出收敛缓慢、对参数敏感甚至不收敛等问题。因此，在工程实践中，迭代法往往需要与对角缩放（Diagonal scaling）、区域分解（Domain decomposition）[67]、多重网格（Algebraic Multigrid ，AMG）[68]等预处理技术相结合，才能在保证数值稳定性的同时发挥其在大规模问题上的性能优势。

从整体上看，线性直接法与迭代法分别代表了“高精度一次性求解”与“低内存渐进逼近”两种不同的数值求解范式。在实际工程求解器中，二者往往并非对立关系，而是通过“直接法用于小规模或作为预处理器，迭代法用于超大规模主要求解”的方式形成互补，共同构成现代CAE求解流程的核心数值基础。

2.2.1.2 线性方程组求解器软件架构

线性方程组求解器的架构可以按照线性直接法和线性迭代法两部分，如图2.22

以及图2.23所示，被划分为三个主要层次：求解器接口层、算法层、公共算法层。每一层都有其特定的功能和职责，共同确保求解器的高效运行和良好的用户体验。

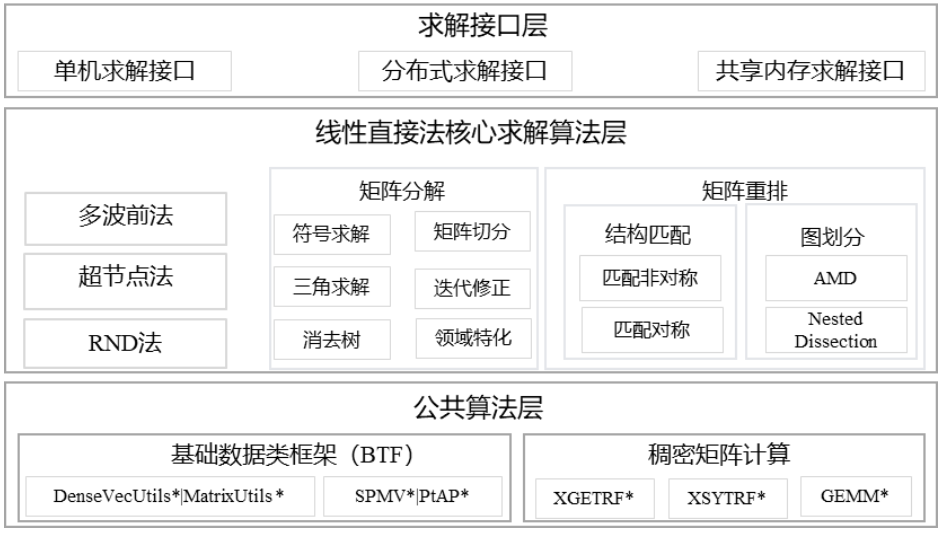

图2.22 线性直接法求解器软件架构

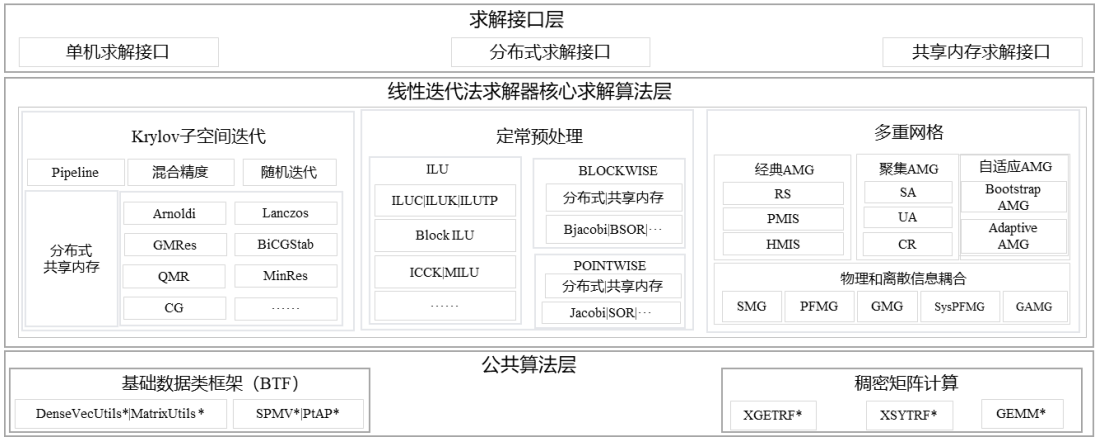

图2.23 线性迭代法求解器软件架构

线性方程组求解器的架构中，求解器接口层作为用户与系统之间的桥梁，承担着接收输入、配置参数和返回结果的核心职责。它提供统一、简洁的调用方式，屏蔽底层算法的复杂性，使用户无需关心具体采用的是直接法还是迭代法。该层还负责对输入矩阵和向量进行合法性校验，封装求解状态与结果，并处理可能出现的异常情况，如矩阵奇异或内存不足，从而确保良好的用户体验和系统的鲁棒性。

算法层是求解器的核心逻辑所在，明确划分为线性直接法和线性迭代法两大分支。直接法部分主要实现如LU分解、Cholesky分解等精确求解策略，适用于中小规模或结构化问题，强调数值稳定性和计算精度；而迭代法部分则包括共轭梯度法、GMRES、BiCGSTAB等Krylov子空间方法，针对大规模稀疏系统设计，通常结合预

处理技术以提升收敛速度。该层根据问题特性动态选择合适算法，并在效率、精度与内存占用之间取得平衡，是决定求解性能的关键环节。

公共算法层为上层各类求解方法提供共享的基础组件和通用工具，避免重复开发并提升整体代码质量。它包含矩阵与向量的基本运算、稀疏存储格式支持、内存管理机制、收敛判断逻辑以及预处理器的统一接口等。这些底层服务不仅支撑了直接法和迭代法的高效运行，还便于集成高性能计算库（如 BLAS、CUDA）或扩展至分布式环境。通过将通用功能下沉到这一层，整个求解器架构实现了高内聚、低耦合的设计目标，增强了可维护性与可移植性。

2.2.1.3 线性求解器整体求解流程

在构建大规模科学计算与工程仿真系统的核心架构时，线性方程组的高效求解往往是决定整体性能瓶颈的关键环节。面对日益复杂的物理模型与海量离散化数据，单一的策略已难以兼顾求解的鲁棒性、收敛速度与资源消耗。因此，构建一个分层化、自适应的线性求解体系显得尤为迫切。本章节将深入剖析线性求解器的整体执行流程，并重点阐述两大核心支柱：一是针对稠密子结构或强耦合区域、旨在通过直接法消除局部自由度的线性超节点求解器，二是面向大规模稀疏系统、凭借低内存占用与高并行潜力实现全局收敛的线性迭代法求解器，下图展示了不同类型问题所推荐选择的求解方法。本节通过对这两类求解机制的协同运作与适用边界进行系统性解读，我们将揭示如何在不同计算场景下实现精度与效率的最优平衡。

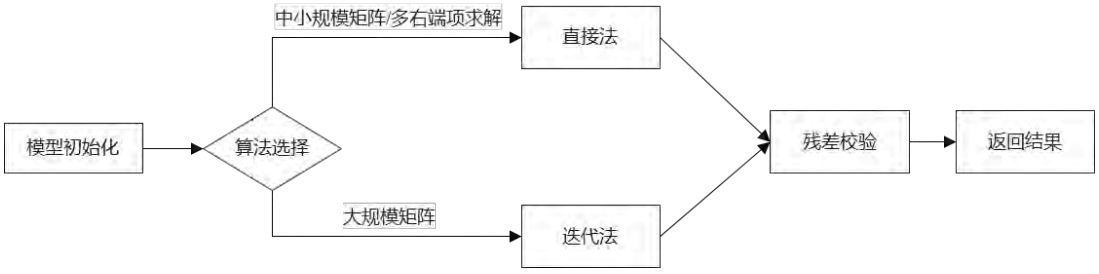

图2.24 线性求解器求解流程

（一）线性超节点直接法整体求解流程

LU 分解适用于一般非对称矩阵，其基本形式为：

$$A = LU$$

其中 L 是单位下三角矩阵，U 是上三角矩阵。分解结束后，即进行三角方程组求解：

$$Ly = b$$

再回带得到原问题的解：

$$Ux = y$$

当矩阵数值条件较差时，通常需要引入主元选取（pivoting）[69]以提升数值稳定性，此时分解形式可表示为：

$$A = PLUQ$$

其中，P 和 Q 分别为行与列的排列矩阵。

对于满足 $A = A^*$，$\forall x \neq 0$，$x^*Ax > 0$ 的共轭对称正定（Hermitian Positive-Definite，HPD）矩阵，其中$()^*$表示矩阵/向量的共轭转置。对于HPD矩阵可以采用不需要主元选取、常数因子更小且数值鲁棒性更好的分解方式。其中，Cholesky分解将矩阵表示为：

$$A = LL^*$$

其中，L 是对角线元素全为正数的下三角矩阵。矩阵 A 能唯一 Cholesky分解的充分必要条件是 A 共轭对称且正定。在更一般的对称但不完全正定情形下，常采用 LDL 分解，将矩阵分解为：

$$A = LDL^*$$

其中 D 为对角矩阵或块对角矩阵，该形式在保持对称结构的同时具备更广的适用范围。

如图2.25的算法流程所示，超节点稀疏直接法通常由分析阶段、数值分解阶段和三角求解阶段三部分组成。分析阶段负责排序、符号分析与超节点数据结构构建，数值分解阶段执行超节点级的矩阵分解，三角求解阶段则基于已分解的因子完成前向与回代计算。本节将介绍上述三个阶段的作用和计算内容，对部分关键技术与实现细节进行说明。

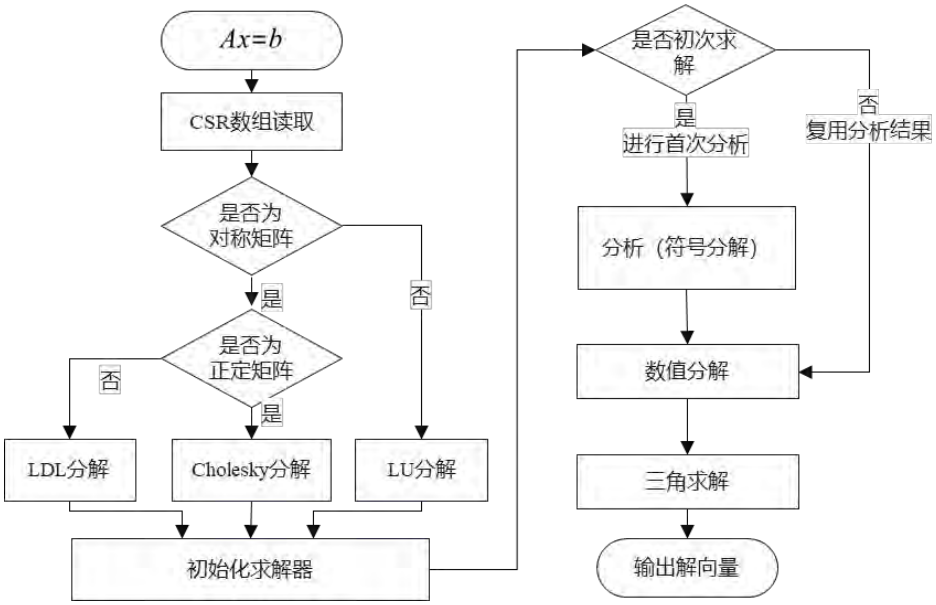

图2.25 超节点稀疏直接法算法流程

a) 分析阶段

分析阶段是稀疏直接法在数值分解之前的关键预处理步骤。其核心目标是在不涉及具体数值计算的前提下，仅基于矩阵的稀疏结构，为后续的数值分解与求解过程构建高效、稳定的数据组织与执行策略。该阶段的优劣将直接影响数值分解的时间复杂度、内存消耗以及并行可扩展性，因而在整个求解流程中具有决定性作用。

从功能上看，分析阶段并不对矩阵元素进行数值运算，而是围绕矩阵的稀疏结构完成一系列结构性决策，包括填充控制、消元顺序确定、符号分解以及数据结构构建等。这些决策一旦确定，通常在同一矩阵对应的多次求解过程中保持不变。

符号分解以及数据结构构建是分析阶段核心目标，稀疏矩阵通常采用压缩稀疏行（Compressed Sparse Row, CSR）或压缩稀疏列（Compressed Sparse Column, CSC）格式进行存储与运算。在该格式中，每一行（或列）的非零元素通过偏移指针定位，而偏移量由对应行（或列）的非零元数量决定。为避免插入写入时的扩容操作，必须通过预先得知分解后各行列的非零元数量并预先分配内存。

填充控制通过矩阵排序（Ordering）实现，是符号分解前的关键优化步骤。由于直接法在消元过程中会引入新的非零元，称为填充元，不当的消元顺序可能导致填充急剧增长，显著增加内存消耗与计算开销。为此，通过对矩阵的行列施加合适的置换（Permutation）来实现排序，以最小化填充总量。

矩阵分解过程中的填充元传播路径则依赖于消去树（Elimination Tree）[70]来确

定，它在实际分解之前揭示了各行（或列）之间的依赖关系，为分解和回代阶段的并行任务划分和调度提供指导。

综上，排序策略、消去树构建与符号分解共同构成了稀疏直接法分析阶段的主要工序，在不涉及任何浮点运算的前提下，为后续高效率、低内存开销的数值分解奠定坚实基础。

b) 分解阶段

在完成分析阶段并确定矩阵排序、消去树以及符号结构之后，即可进入数值分解阶段。数值分解阶段的主要任务是在既定的稀疏结构约束下，对矩阵执行实际的数值计算，生成用于求解的因子矩阵。该阶段通常是稀疏直接法中计算量最大的部分。

对于一般非对称矩阵，在引入行、列排序的情况下，数值分解可表示为：

$$PAQ = LU$$

其中 P 和 Q 为排列矩阵，L 为单位下三角矩阵，U 为上三角矩阵。对于对称，则可采用LDL或Cholesky分解形式，以进一步降低计算复杂度和存储需求。

与稠密直接法不同，稀疏直接法中的数值分解严格受限于分析阶段确定的非零结构。仅在符号分解阶段预测为非零的位置，才会在数值分解阶段分配存储并参与计算。这一特性使得分解的计算复杂度与因子矩阵的非零元数量相关，而不再与矩阵维数的三次方成正比。

从计算流程上看，超节点数值分解通常包含两个相互交替的阶段：一方面对当前超节点对应的稠密面板执行分解操作，另一方面将分解结果更新到其后继超节点所对应的矩阵块中。超节点之间的执行顺序及其依赖关系由消去树所刻画，使得数值分解过程在保持正确性的前提下，天然具备分阶段、可调度的结构特征，为串行优化和并行执行提供了基础。

c) 三角求解阶段

在完成数值分解之后，线性方程组的求解通过三角求解（回代）阶段完成。该阶段基于已生成的因子矩阵，对右端项执行前向与回代计算，从而得到原始线性系统的解向量。

对于LU分解形式，回代过程通常分为两个步骤。首先求解下三角线性方程组

$$Ly = Pb$$

随后求解上三角线性方程组

$$Ux = y$$

其中 b 为原始右端项，x 为最终解向量。由于分解结果可以被重复使用，对于同一个系数矩阵，当存在多个右端项时，只需执行一次数值分解，而回代阶段可以针对不同右端项高效重复执行。

在稀疏直接法中，回代阶段的计算同样受到因子矩阵稀疏结构的约束，其访问模式由 L 和 U 的非零分布所决定。尽管单次回代的计算量通常低于数值分解阶段，但在大规模问题或多右端项场景下，其累计开销仍可能占据相当比例，因此同样需要精心设计。

在超节点方法中，回代阶段同样可以在超节点粒度上进行组织。同一超节点内的多列或多行可被视为稠密块，通过将多个向量级操作合并为矩阵级运算，不仅可以提升缓存利用率，还能够减少调度与函数调用开销。结合消去树所提供的依赖关系，超节点级回代过程可以实现有序执行，并在条件允许时支持并行调度。

（二）线性迭代法整体求解流程

与线性直接法通过矩阵分解一次性获得精确解不同，线性迭代法是一类通过构造逐步逼近过程来求解线性方程组的方法。其基本思想是：从一个初始猜测 x^0 出发，通过反复迭代更新：

$$x^{(k+1)} = F(x^k)$$

使得 x^k 逐步收敛到真实解 x。从数学角度看，迭代法并不显式构造矩阵 A 的逆或分解形式，而是通过矩阵-向量乘和向量运算逐步消除残差

$$r^k = b - Ax^k$$

当残差范数满足预设精度要求时 $||r^k|| < \epsilon$，即认为迭代收敛，此时的 x^k 可以被认为是达到精度要求的近似的真实解。

在CAE、计算流体力学、电磁场、结构分析等工程仿真中，离散化后的线性系统往往具有如下特征：

1) 问题规模极大，自由度可达百万至上亿；

2) 矩阵高度稀疏：非零元比例极低；

3) 物理问题主导谱特性：如椭圆型PDE、对流扩散问题等；

4) 单次求解即可：通常只需求解一次或少量右端项。

在这单次求解场景下，尽管稀疏直接法在数值稳定性和鲁棒性方面具有明显优势，但其工程代价逐渐变得难以接受：分解阶段内存消耗巨大，常成为瓶颈；填充随问题规模快速增长；分解过程并行效率受限（尤其在多节点环境中）。这催生了对内存占用更低、并行扩展性更好的迭代法的广泛需求。

以下表格展示了直接法和迭代法各自的特性：

表2.2 直接法与迭代法特性对比

特性	**直接法（稀疏LU/Cholesky）**	**迭代法（如Krylov子空间方法）**
基本思想	矩阵分解，解三角方程组	从初始猜测出发逐步逼近解
内存使用	较高（需存储分解后的因子）	较低（只需存储原矩阵和向量）
计算复杂度	分解成本高，但每右端项求解成本低	每步迭代成本低，但迭代次数不定
问题规模	中小规模到中大型（千万自由度以下）	超大规模问题（自由度可达数十亿）
稳定性	较稳定，但可能受舍入误差影响	对条件数敏感，常需预处理改善收敛性
并行潜力	有限（虽然多前沿方法等可并行化）	较高（矩阵向量乘、向量运算易并行）
适用矩阵类型	一般非奇异矩阵（对称时可选用LDL/Cholesky）	需要矩阵对称正定（如共轭梯度法）或至少非奇异（如GMRES、BiCGSTAB）
精确度	机器精度解（不考虑舍入误差）	可控制近似解精度

通过表2.2，迭代法的具体优势总结如下：

1) 内存可扩展性：迭代法通常只需原矩阵存储（保持稀疏性）和若干工作向量，内存占用小。

2) 计算灵活性：迭代法可在达到指定精度（如相对残差 $||r^k||/r^0 < \epsilon$）后停止，这在许多工程应用中足够，且能节省大量计算时间。

3) 适合并行与分布式计算：迭代法的核心操作是矩阵-向量乘、向量内积更新，这些操作易于在CPU/GPU集群上高效并行，扩展性良好。

4) 可与问题物理特性结合：迭代法（特别是多重网格法、区域分解法）能利用微分方程本身的数学特性，设计出具有最优或近似最优复杂度 $O(n)$ 的算法。

虽然迭代法有种种优势，但迭代法最大的问题是本身对病态问题收敛缓慢。而迭代法中最关键的技术—预处理技术（Precondition），不仅可以克服这一问题，甚至可以普遍提高迭代法的求解效率。

具体而言，预处理技术首先构造一个近似 A^{-1} 的算子 M^{-1}，使预处理后的系统

$$M^{-1}Ax = M^{-1}b$$

条件数大大改善，从而大大减少迭代步数，进而加速收敛。常见的预处理可以按照如下划分：

1) 不完全分解类：不完全LU分解（Incomplete LU Factorization ，ILU）、不完全乔列斯基分解（Incomplete Cholesky Factorization ，ICC）等；

2) 代数预条件子：雅克比预条件子（Jacobi）、高斯-赛德尔预条件子（Gauss-Seidel ，GS）、逐次超松预条件子（Successive Over-Relaxation ，SOR）、对称逐次超松弛预条件子（Symmetric Successive Over-Relaxation ，SSOR）；

3) 多重网格类：代数多重网格（AMG）、几何多重网格（GMG，Geometric Multigrid）；

4) 块预条件子：块雅克比预条件子（Block Jacobi）、块高斯-赛德尔预条件（Block GS）、块不完全LU分解（Block ILU）。

其中 ILU、Jacobi和AMG是工程中最常见、最通用的几类预条件子。在并行计算中，Block Jacobi、Block ILU等是构建并行预条件子的基石。它们通常将一个全局问题分解为子域问题，子域内部可使用ILU或直接法高效求解。

以上预条件子中，ILU和AMG在工程中的应用最广泛。其中，ILU因其通用性强，实现相对简单，是许多求解器的默认或基础选项，它也常作为其他复杂预条件子的组成部分。AMG针对“椭圆”性问题（如压力泊松方程），具有接近 $O(n)$ 的复杂度，收敛速度对网格尺寸不敏感，且尤其适用于大规模扩散、结构力学问题。此外，AMG还能够实现混合精度加速，例如面向GPU的异构加速技术可以对AMG算法进行深度适配和优化[71]，获得显著加速比。

雅克比（Jacobi）预处理子是最简单、计算成本最低的预条件，仅需对原矩阵取对角元求逆。在评估更复杂算法（如ILU或AMG）时，通常会以Jacobi预条件子的性能作为比较的基准线（Baseline）。如果能被Jacobi显著改善，说明原矩阵对角占优性尚可；如果Jacobi效果不佳，则暗示问题具有病态性，需要更强力的工具。当方程组的对角元包含了问题物理量的主要信息（例如，在有限元离散中，对角元与局部刚度或质量相关），此时预处理后的矩阵性质往往能够得到改善，求解效率

能够显著提高。如果Jacobi预条件效果甚微，那通常意味着问题的耦合性强，或离散导致的条件数恶劣，此时可以考虑采用更高级的预条件子。

根据流程上，迭代法主要可以分为初始化问题定义阶段、预处理子构建阶段、初始解选择阶段、循环迭代阶段和残差计算与收敛判定阶段。具体可以参考图2.26的流程。

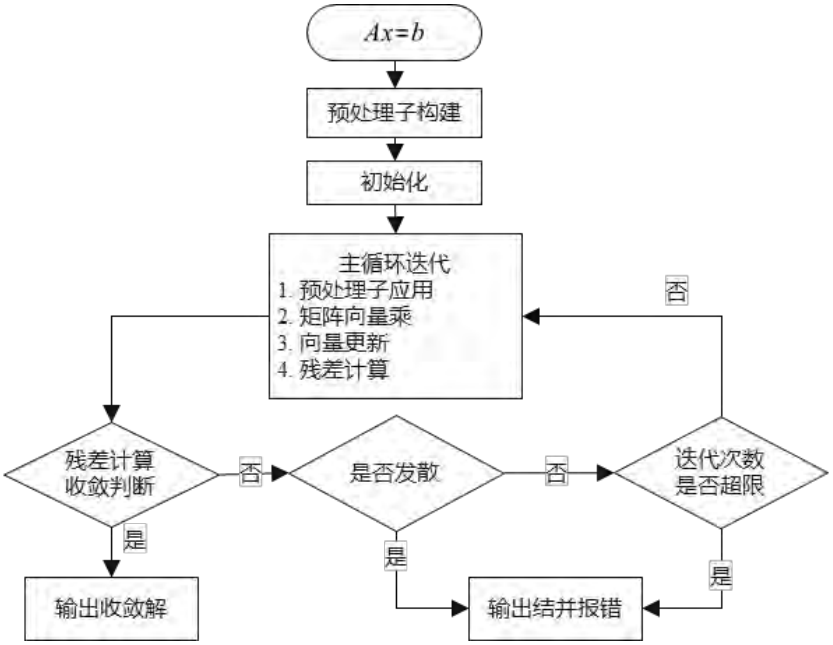

图2.26 线性迭代法算法流程

a) 初始化阶段

初始化阶段的目标是为迭代求解建立一个正确的起点和必要的工作环境。此阶段不涉及核心的迭代循环，但它的正确性与效率直接影响后续迭代的收敛行为。

这一阶段主要是进行问题设置和参数设置：系数矩阵 $A \in \mathbb{R}^{n\times n}$，右端向量 $b \in \mathbb{R}^n$， $x \in \mathbb{R}^n$。参数配置：用户根据问题特性和经验，选择具体的Krylov算法（如GMRES，CG，BiCGSTAB等），并设定算法控制参数，主要包括：收敛容差 ϵ（例如，基于相对残差：$||r^k|| \,/\, ||r^0|| < \epsilon$）、最大迭代次数 N_{max} 和重启参数（针对GMRES等重启算法）等。

b) 预处理子构建阶段

预条件子 M^{-1} 的构造是提升算法收敛性能最为关键的前处理步骤，其目标是使预处理后矩阵 $M^{-1}A$ 的谱分布更为聚集，从而显著加速Krylov子空间的收敛速度。构造策略具有高度的矩阵依赖性：

下面以 Jacobi 预处理子为例，进行简介：

将系数矩阵 A 分解为对角部分与非对角部分：

$$A = D + (A - D)$$

其中 $D = diag\{a_{11}, a_{11}, \dots, a_{nn}\}$ 表示 D 的对角矩阵。Jacobi预处理选取对角矩阵

$$M = D$$

可以看到，对角矩阵易于存储（可以直接以向量的形式）；逆算子 D^{-1} 的计算代价极低；在一定程度上可以平衡不同变量的尺度。使用预处理后的问题变为：$M^{-1}Ax = M^{-1}b$。

Jacobi预处理为代表，预处理子的构建本质上是在计算代价可控的前提下，用一个易于求逆的近似算子替代原矩阵参与迭代过程，从而在不显著增加单步计算成本的情况下显著改善整体收敛性能。

c) 迭代构造与求解阶段

这是Krylov方法的核心循环。其本质是：在每一步迭代 k 中，扩展Krylov子空间，并在该子空间内寻找一个满足某种最优准则的近似解。

该阶段的核心数学框架可统一描述如下：

对于线性系统 $Ax = b$，给定初始猜测 x^0 和初始残差 $r^0 = b - Ax^0$，第 k 步迭代的Krylov子空间定义为：

$$K_k(A, r_0) = (A, r_0) = span\{r^0, Ar^0, A^2r^0, \dots, A^{k-1}r^0\}.$$

Krylov方法寻求形如 $x^k = x^0 + z^k$ 其中的解，其中 $z^k \in K_k(A, r_0)$。不同方法通过施加不同的最优化准则（如残差最小化或Galerkin正交性）来唯一确定 z_k。

尽管不同Krylov方法在实现细节上各异，但其核心循环可抽象为以下四个连贯的步骤：

1) 子空间扩展与正交基构造：在为 K_k 中进行数值稳定的计算，需要构造一组标准正交基 $V_k = [v_1, v_2, \dots, v_k]$。通用的过程包括Arnoldi，Lanczos，这里不展开介绍。

2) 子空间投影与约化问题形成，如GMRES：

$$\min\left|\left|b - A(x^0 + V_k y)\right|\right| = \min\left|\left|\beta e_1 - H_k y\right|\right|,\ \beta = ||r^0|$$

3) 问题求解：求解上一步形成的低维稠密问题，得到子空间中的最优坐标。

GMRES需要求解一个最小二乘问题（通常使用Given旋转迭代更新其QR分解来实现）；CG则通过一组三项系数递推公式隐式地获得解，这些系数由当前残差、搜索方向与矩阵作用结果的内积决定。

4) 近似解与迭代状态更新：第 3) 步的求解，更新原问题的近似解和算法状态，为下一步迭代或收敛判断做准备。

d) 收敛判断

每步迭代后，计算当前近似解 x^0 的残差范数。对于像GMRES这样的方法，最小二乘问题的残差可以从 H_k 和 y 直接、廉价地计算，而无需显式计算 $b - Ax$，这极大地提高了效率。最后判断是否满足：$||r^k|| \leq \epsilon$ 或达到 N_{max}.

此阶段循环进行，直到满足收敛条件或迭代失败。成功的迭代法，其残差范数 $||r^k||$ 通常会随着 k 的增长而单调或近似单调下降，直至达到目标精度，此时的迭代解 x^k 可以被认为满足精度的解。

2.2.1.4 线性求解器关键技术

（一）直接法：减少填充排序

在稀疏矩阵分解中，填充规模直接决定计算量与内存占用，是影响求解效率的关键因素。影响填充的主要因素之一是消元顺序。以图2.27和图2.28所示的箭头形矩阵为例，若按照原始方程与变量顺序进行消元，则在消去中心变量后，其余变量之间原本不存在的耦合关系将被引入。以下面经典的箭头矩阵举例：随着消元过程的推进，这些新增的耦合会不断向矩阵的其余部分传播，最终导致分解得到的 L 和 U 因子在结构上几乎完全稠密。

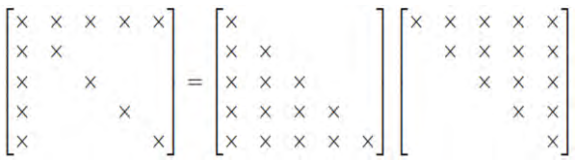

图2.27 没有重排的箭头矩阵的分解结果

但如果在分解前对原矩阵的行和列进行逆序排列，则消元过程中的结构演化将发生根本性变化，没有产生任何额外的填充。

图2.28 行列重排后箭形矩阵的分解结果

这一对比清楚地表明，同一个稀疏矩阵的不同排序，其分解后的非零结构可能呈现出完全不同的形态：从近似稠密到严格保持稀疏。基于上述原因，稀疏直接法在工程实现中通常不会采用矩阵的自然顺序进行消元，而是通过专门的策略，在分解前对变量顺序进行系统优化，通过对矩阵进行适当的行列排序，以降低分解过程中产生的填充规模。

对原矩阵进行行列排序，本质上等价于对方程与变量重新编号，改变分解过程中的校园顺序。这一预处理过程通常称为矩阵排序。部分资料也将此方法称作重排序（reordering）。合理的排序策略往往能够将分解后的非零元数量降低一个数量级以上，保证了稀疏直接法能否在时间与空间上保持可扩展性，是分析阶段最为关键的结构性决策之一。

对于填充元的产生一般采用图论分析[72]，如图2.29所示，我们定义对应的无向图$G = (V, E)$：

1) 顶点集合（Vertices）V：对应矩阵的行号（或列号）。如果矩阵是 $n \times n$，即有 n 个顶点。

2) 边集合（Edges）E：对应矩阵中的非零元素，如果矩阵元素 $A_{i,j} \neq 0$ (且 $i \neq j$)，顶点 i 和顶点 j 之间有一条边，两个顶点是邻居。

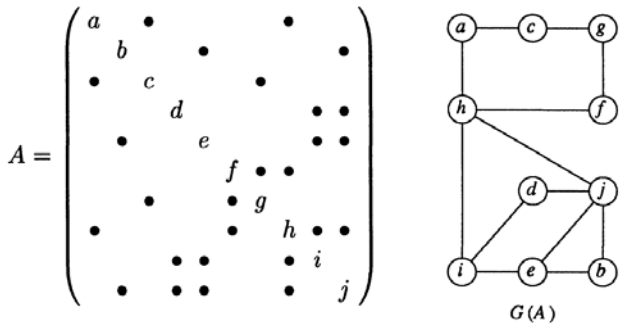

图2.29 矩阵A及其对应的图G(A)

矩阵分解的过程在图论中等价于：

1) 选择一个顶点 v 进行消除（对应矩阵的某一行/列作为主元）。

2) 填充步骤：将 v 的所有邻居节点两两相连。如果邻居之间原本没有边，这就加了一条新边（这就是填充）。

3) 从图中移除 v 及其相连的边。

4) 重复上述步骤直到图为空。

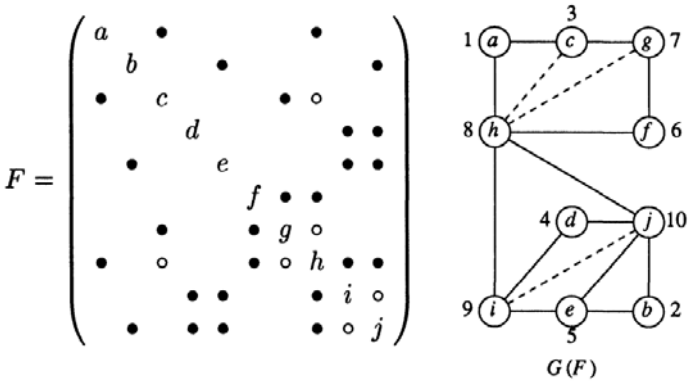

图2.30 矩阵F及其对应的图G(F)

图2.30展示了原矩阵分解后的新矩阵 $\mathrm{F} := \mathrm{L} + \mathrm{U}$ 和对应的图 $G(F)$，其中空心圆表示填充，对应图中的虚线边。可以看到有3个虚线边产生，即此分解产生了三个填充元。

假如一个节点连接了 k 个邻居，消除它最坏情况可能会导致 $k(k-1)/2$ 条新边的产生。因此，对于排序消除顺序的直观认识是：应当先消除那些邻居很少的节点。但理论分析寻找使填充最小的最优排序是NP难问题，在实际应用中一般依赖启发式方法获得近似最优解。常用排序方法包括基于局部优化的最小填充（Minimum Fill，MF）和最小度（Minimum Degree，MD）策略[73]，以及基于分治思想的嵌套剖分（Nested Dissection，ND）方法及其混合形式[74]。

（二）直接法：消去树构建和符号分解

符号分解（Symbolic Factorization）阶段是指在不引入具体数值运算的前提下预先确定分解后的矩阵非零元位置。消去树描述了不同列（或超节点）在消元过程中的先后关系，并为矩阵分解后的非零结构的预测提供了清晰的层次化约束。

在工程实现中，消去树主要用于三类任务：其一，用于确定符号分解过程中填充元的传播范围；其二，用于指导分解后结构与存储布局的构建；其三，为后续并

行任务调度和超节点划分提供依赖关系基础。正是由于消去树在结构分析中的核心作用，其已成为现代稀疏直接法分析阶段不可或缺的基础工具。

在进行符号分解之前，消去树通常需要进行后序排序（Postordering）。该排序属于一种拓扑排序操作，不改变消去树所描述的消元依赖关系，也不会影响填充模式，仅对LU因子的行列编号方式进行调整。通过后序排序，可以使同一超节点内的列在编号上保持连续，从而简化数据结构设计，并提高后续超节点级数值分解的执行效率。

在获得经过后序排序的消去树之后，即可基于消去树开展符号分解。根据分析粒度的不同，符号分解通常可分为逐元素的非超节点符号分解和逐块的超节点符号分解。实际工程实现中，通常采用超节点符号分解方式，以更好地匹配稠密块计算模型与并行执行需求。

符号分解阶段通常在不引入具体数值运算的前提下进行，并普遍采用无数值抵消假设，即假定非零元参与运算后产生的结果仍视为非零。该假设一方面显著简化了符号分析过程，另一方面在工程实践中具有良好的合理性：数值抵消现象较为少见，且即使发生，也难以在分析阶段准确预测并加以利用。根据结构粒度的不同，符号分解可分为逐元素的非超节点符号分解和逐块的超节点符号分解。

在超节点稀疏直接法中，符号分解不仅用于确定LU因子的非零结构，还承担着构建超节点结构的重要职责。所谓超节点，是指在 L 因子中具有相同或高度相似非零结构的连续若干列，这些列可以被组合成稠密子矩阵，并通过高效的BLAS算子统一处理，从而显著提升数值分解阶段的计算效率。

（三）直接法：匹配预处理

在稀疏直接法中，数值分解的稳定性不仅与矩阵的整体条件数有关，也与主对角线附近元素的分布密切相关。对于实际工程问题生成的稀疏矩阵，若大数值元素远离对角线，直接进行LU或LDL分解往往需要频繁引入主元选取（Pivoting），从而破坏分析阶段所确定的结构假设，增加填充并降低整体性能。

为改善这一问题，求解器在分析阶段引入基于匹配（Matching）的数值稳定性预处理。该预处理的核心目标是在不改变线性方程组解的前提下，通过行列排序，使矩阵中较大的数值元素尽可能靠近对角线，从而提高后续数值分解的稳定性并减少对动态主元交换的依赖。

匹配方法可以被视为在矩阵的非零结构上寻找一组行列配对关系，使得被匹配的元素在数值上尽可能“大”，并被排列到对角线位置。该思想与传统减少填充排序不同：排序主要关注填充元控制和结构优化，而匹配侧重于数值尺度的改善。二者在分析阶段承担着互补的角色，共同为数值分解创造有利条件。

对于数值较为病态的矩阵，尤其是对角线上存在 0 的矩阵，可以在排序前添加最大权匹配（Maximum Weight Matching）预处理，提升数值稳定性。不同于排序和符号分解，匹配除了需要系数矩阵的结构信息，还需要数值信息。

对于非对称矩阵 A，匹配预处理如下（以最大权重匹配类型为例）：

$$A \leftarrow P_r D_r A D_c$$

其中 P_r 是行排列矩阵，D_r 和 D_c 是行放缩对角矩阵和列放缩对角矩阵。经过行排序和行列放缩，A 的对角元绝对值均为1，非对角元绝对值均小于等于1。

(1) 静态选主元方法

在LU分解过程中，通过预先构造置换矩阵 P_M（精度排序矩阵）来优化数值稳定性。该过程旨在寻找最优排列使重排后的主元绝对值最大化。在实际求解链中，通常先进行精度排序以确保主元强度，随后执行减少填充排序 P_{Fill}；后者采用对称置换形式 $P_{Fill}P_M A P_{Fill}^T = LU$，以在降低计算复杂度的同时避免破坏已选定的主元位置。NCS 求解器支持以下两种目标函数寻优：

1) max_prod: 最大化主元绝对值的乘积 $\max_{\sigma} \prod_{i=1}^{n} |a_{i\sigma_i}|$

2) max_sum: 最大化主元绝对值之和 $\max_{\sigma} \sum_{i=1}^{n} |a_{i\sigma_i}|$

图2.31 重排流程示意图

如图2.31，$P_M A$，行排序矩阵，P_M的第一行对应第4个元素非零，因此把第4行交换到新矩阵的第1行。

主元乘积的最大化问题可以转化为最小化二分图匹配问题。给定稀疏方阵 A，寻找排列 σ 最大化主元乘积 $\max_{\sigma} \prod_{i=1}^{n} |a_{i\sigma_i}|$，等价于最小化问题 $\sum_{i=1}^{n} c_{i\sigma_i}$，其中，

$a_j = \max_i |a_{ij}|$ 是方阵 A 第 j 列的最大非零数值

$$c_{ij} = \begin{cases} loga_j - log|a_{ij}|, a_{ij} \neq 0 \\ \infty, \quad otherwise \end{cases}$$

以上最小化问题等价于二分图上的最小权重匹配（Minimum Weight Matching）问题。

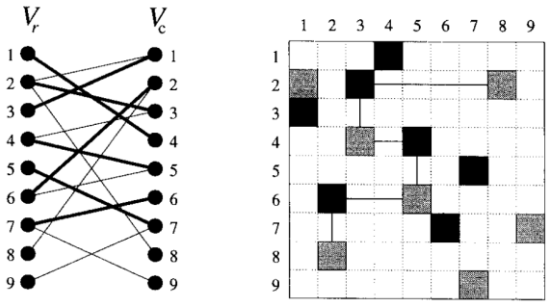

图2.32 最小权重匹配示意图

如图2.32，边 $1 \rightarrow 4$ 表示第 1 行匹配到第 4 列，即选中 $a_{1,4}$ 作为第 1 个主元。因此，静态主元选取方法可表述为：通过构造相应的对偶问题，将主元选取转化为一个最小权重匹配问题，并采用基于增广路的完美匹配（Perfect Matching）算法进行求解。

(2) 主元归一化

$D_r A D_c = A'$，其中 D_r 为 D_r 为行缩放（Scale）对角阵，D_c 为 D_c 为列缩放对角阵，排序后 $A^* = P_M A'$ 主元 A' 主元归一化：所有主元满足 $|A^*(i,i)| = 1, \forall i$，非主元满足 $|A^*(i,j)| \leq 1, \forall i \neq j$

设对偶变量 u, v，满足:

$$\begin{cases} u_i \geq 0, \ \forall i \\ v_j \geq 0, \ \forall j \\ c_{ij} - u_i - v_j \geq 0, \ \forall i,j \end{cases}$$

根据对偶变量u, v 构造 Scaling 矩阵:

$$D_r(i,i) = \exp(u_i), \ D_c(j,j) = \frac{\exp(v_j)}{a_j}$$

$$|A'(i,j)| = D_r(i,i) \cdot |a_{ij}| \cdot D_c(j,j) = \exp\left(u_i + \log\left(|a_{ij}|\right) + v_j - \log(a_j)\right)$$

$$= \exp\left(u_i + v_j - c_{ij}\right) \leq 1$$

$$(i,j) \in M \rightarrow u_i + v_j - c_{ij} = 0 \rightarrow |A'(i,j)| = 1$$

病态矩阵处理方法：

1) 非零元 a_{ij} 绝对数值太小导致 $log\,|a_{ij}|$ 运算误差大：设定阈值特殊处理。

2) 近似“零”行：某一行最大非零元绝对数值很小（$< 10^{-16}$）。解决方案：预缩放给（Prescale）策略，预先进行“行归一化”。

3) 绝对零行：某一行数值全为 0。解决方案：小数 $\sim 10^{-19}$ 替换。

（四）直接法：超节点的识别

在超节点稀疏直接法中，超节点（Supernode）是数值分解阶段的基本计算单元，其核心思想是将具有相同或高度相似非零结构的多列合并为一个整体，以稠密矩阵块的形式进行计算，从而显著提升数据局部性并充分利用高性能BLAS运算内核。在 L 因子中，若连续若干列同时满足以下结构特性，则可构成一个超节点：

1) 对角块的下三角部分完全稠密；

2) 对角块以下部分的非零结构在这些列之间完全相同。

如图2.33所示，若第7－9列满足上述条件，则可将其识别为同一个超节点。对于列数大于 1 的超节点，可在其对角块的严格上三角部分人工填充少量非零元，从而使整个超节点对应的非零结构形成一个二维稠密矩阵块。尽管该过程会引入少量额外填充，但通过将后续运算转化为稠密矩阵计算，能够显著提高数值分解阶段的效率，其收益通常远大于由额外填充带来的开销。

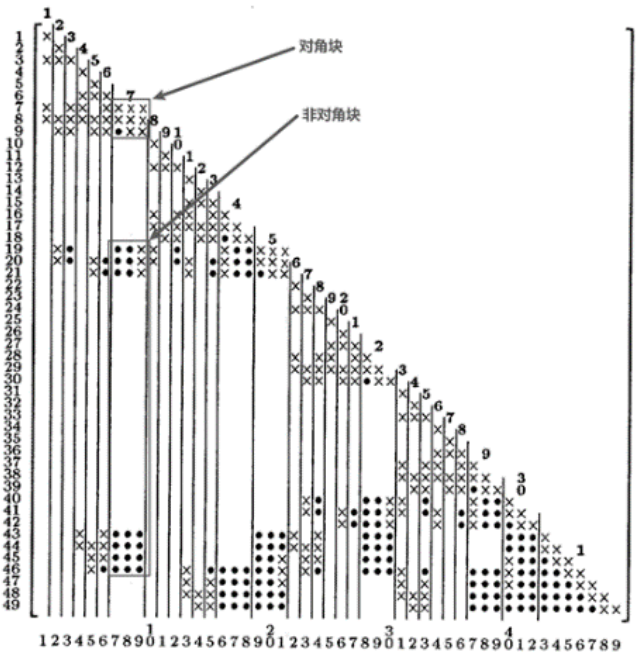

图2.33 超节点识别示意图

根据结构精确性与计算效率之间的不同取舍，工程实现中常将超节点划分为以下几类：

(1) 基本超节点（Fundamental Supernode）

基本超节点的识别强调结构依赖的严格性，其核心判据可以概括为以下两点：

1) 结构一致性条件

对于候选的相邻列集合，这些列在 L 因子中必须具有完全相同的非零模式（除对角块内部的下三角部分外）。这保证了将其合并为一个超节点后，不会引入额外的结构差异或不必要的数值更新。

2) 消去树约束条件

候选列在消去树中的依赖关系必须保持一致。具体而言，构成同一基本超节点的列在消去树中应位于同一子树路径上，且其在消去树中的父子关系不允许跨越不同子树分支。该约束确保了不同基本超节点之间的更新依赖严格受消去树层次所限制，从而避免在数值分解阶段产生隐式的数据依赖。

满足上述条件的列集合可以安全地合并为一个基本超节点。由于其依赖关系完全由消去树刻画，不同子树中的基本超节点在数值分解阶段不存在相互更新，因此基本超节点在并行调度中具有最清晰、最严格的依赖语义，常被视为理论意义上的“最小安全单元”。

(2) 极大超节点（Maximal Supernode）

极大超节点在结构一致性方面与基本超节点具有相同的要求，但在消去树约束上采取了更为宽松的策略，从而形成更大的稠密块，但其代价是可能引入额外的超节点间更新操作。其识别过程遵循以下原则：

1) 在保持结构一致性的前提下，尽可能将相邻的列合并为更大的列集合；

2) 制要求候选列在消去树中严格满足基本超节点的父子路径约束。

这种策略的直接结果是形成规模更大的稠密块，从而在数值分解阶段提高单次 BLAS 运算的计算强度。然而，其代价是可能在不同超节点之间引入额外的更新操作，使得依赖关系不再完全由消去树的子树结构所隔离，适用于对单节点计算性能要求较高、并行粒度相对粗的场景。

(3) 松弛超节点（Relaxed Supernode）

松弛超节点通过人为放宽结构一致性条件，将原本不完全满足定义的列合并

为一个超节点，以进一步扩大稠密块规模。松弛超节点并不存在统一的理论定义，其松弛准则通常由算法设计者根据工程需求自行设定。在当前实现中，松弛超节点的识别主要遵循以下思路：

1) 允许候选列之间存在有限的结构差异；

2) 当结构差异小于给定阈值时，将其合并为同一超节点；

3) 通过引入少量额外填充，换取更大的稠密块规模。

（五）直接法：映射

在并行超节点直接法中，映射（Mapping）是指将分析阶段得到的超节点集合，按照一定策略分配到并行计算资源上的过程。映射结果决定了超节点由哪个进程负责其数值分解、更新与回代计算，并由此确定了计算负载的分布方式以及跨进程通信的模式。因此，映射是连接算法结构与并行执行环境的关键环节，对整体性能和可扩展性具有决定性影响。

在超节点直接法求解器中，超节点既是数值分解与回代阶段的基本计算单元，也是并行调度与数据分布的最小粒度对象。围绕“如何在保证数值正确性与结构依赖不被破坏的前提下，实现稳定、低开销且可预测的并行执行”，求解器对映射、通信装箱优化以及性能建模进行了系统性设计。

1) 基于消去树层级的超节点静态映射策略

NCS采用基于消去树层级（L_n）的超节点静态映射策略。所谓消去树层级，是指超节点在消去树中的层级或深度位置，该信息在分析阶段即可确定。消去树层级静态映射的基本思想是：在分析阶段一次性确定每个超节点所属的MPI进程，使映射结果与消去树的结构依赖保持一致，而不依赖运行阶段的动态任务迁移或重分配。

具体而言，消去树层级映射利用消去树的层次结构，将位于相同或相近层级的超节点映射到同一MPI进程或相邻进程上。由于消去树天然刻画了超节点之间的更新依赖方向，该映射方式能够在保证数值正确性的前提下，使更新依赖在进程间的分布更加集中，从而减少跨进程更新的频率与通信范围。

与完全动态映射策略相比，消去树层级静态映射避免了运行阶段频繁调整任务归属所带来的调度开销，具有执行行为稳定、结果可复现、性能可预测等工程优势。同时，由于映射结果在分析阶段即可确定，也为后续通信优化与性能建模提供

了清晰的结构基础。

2) 更新阶段的通信装箱优化

在超节点数值分解的更新阶段，多个超节点往往需要向同一目标进程发送更新数据。若对每个超节点分别进行通信，不仅会产生大量小规模消息，还会导致频繁的装箱（Packing）操作，从而显著放大通信启动延迟并降低整体效率。

针对这一问题，NCS求解器在消去树层级静态映射的基础上引入通信装箱优化策略。该策略以映射结果为依据，将来自同一源进程、目标进程相同且在执行顺序上相邻的多个超节点更新数据合并到一个连续缓冲区中，通过一次通信操作完成发送。通过减少装箱次数和通信启动次数，装箱优化能够有效降低小消息比例，并显著摊薄通信相关的固定开销。

需要强调的是，该装箱优化仅作用于通信层面，不改变超节点之间的数值依赖关系，也不影响分解结果的正确性。结合消去树层级静态映射后，更新通信模式呈现出更强的规律性，使得装箱与合并操作更易实施，也更具稳定性。

（六）直接法：迭代修正

为提升直接法求解所得初始解的数值精度，并在采用静态主元或阈值主元策略时增强整体鲁棒性，NCS在数值分解完成后引入迭代修正（Iterative Refinement, IR）过程。

在第k次迭代中，迭代修正的基本形式为：

$$r_k = b - A\,x_k, \qquad A\Delta x_k = r_k, \qquad x_{k+1} x_k + \Delta x_k,$$

其中线性系统 $A\,\Delta x_k = r_k$ 直接复用已计算好的分解因子进行求解，因此单次修正的计算代价远低于重新分解矩阵。迭代修正过程中，解的质量通过后向误差（Backward Error）进行评估。

对当前近似解$\bar{x}$，首先计算逐分量缩放残差。对于一般情形，定义

$$\eta_i = \frac{|b - A\,\bar{x}|_i}{(|b| + |A||\,\bar{x}|)_i},$$

对于少数分母过小但分子非零的“例外方程”，采用改进的缩放形式：

$$\eta_{\mathrm{i}} = (|\mathrm{b} - \mathrm{A}\bar{\mathrm{x}}\,|_{\mathrm{i}})/((|\mathrm{A}||\bar{\mathrm{x}}|)_{\mathrm{i}} + \|(\mathrm{A_i})_\infty\|(\bar{\mathrm{x}})_\infty),$$

其中 A_i 表示矩阵 A 的第 i 行。

为了衡量误差在每个分量上的相对重要性，计算权重向量 w_1（支持鲁棒性处

理）：

$$w_1 = |A|\,|x^{(k)}| + |b|$$

为防止分母为零，常采用分量形式：

$$w_{1,i} = \max(\,|(Ax^{(k)})_i|, |b_i|, \epsilon)$$

其中 ϵ 为机器精度或极小常数。

1) 估计稀疏向后误差 $\eta^{(k)}$

根据 Arioli 准则，估计当前迭代的向后误差：

$$\eta^{(k)} = \|r * w_1\|_\infty / \|(Ax^{(k)}) * w_2\|_\infty$$

注：这里*表示逐元素除法（Hadamard Division）。w_2 通常设为单位向量或矩阵 A 的列缩放向量。

2) 收敛判断与解的更新

收敛检查：如果 $\eta^{(k)} \le \eta_{tol}$，则认为精度已达标，停止迭代。

执行修正：若未收敛，利用已有的因式分解结果求解修正方程：

$$Ad = r$$

更新解：

$$x^{(k+1)} = x^{(k)} + d$$

（七）直接法：并行求解框架

超节点直接法求解器采用基于超节点的并行稀疏直接法作为核心求解框架，其整体并行化设计并非以算子或数值核为中心，而是以超节点到并行计算资源的映射为主线展开。如图2.34所示，在该方法中，超节点既是数值分解与回代阶段的基本计算单元，也是并行调度、数据分布以及通信组织的最小粒度对象。

在分析阶段完成矩阵排序、符号分解与超节点构建之后，求解器首先基于消去树结构确定超节点之间的依赖关系，并据此执行静态映射：将超节点一次性映射到MPI进程上，使映射结果与消去树的层级关系保持一致。该映射在分析阶段确定，在数值分解与回代阶段保持不变，从而避免运行过程中频繁的任务迁移与动态调度开销，保证执行行为的稳定性与可预测性。

在数值分解阶段，每个MPI进程负责其本地超节点的面板分解及相应的更新计

算，跨进程的数值依赖通过超节点级更新进行显式通信。由于映射已在结构层面尽量集中依赖关系，更新通信具有较强的规律性，可进一步通过通信装箱与合并策略减少小消息数量，降低通信启动成本。节点内则充分利用超节点内部的稠密块结构，通过共享内存并行和高效BLAS运算提升单节点计算效率。

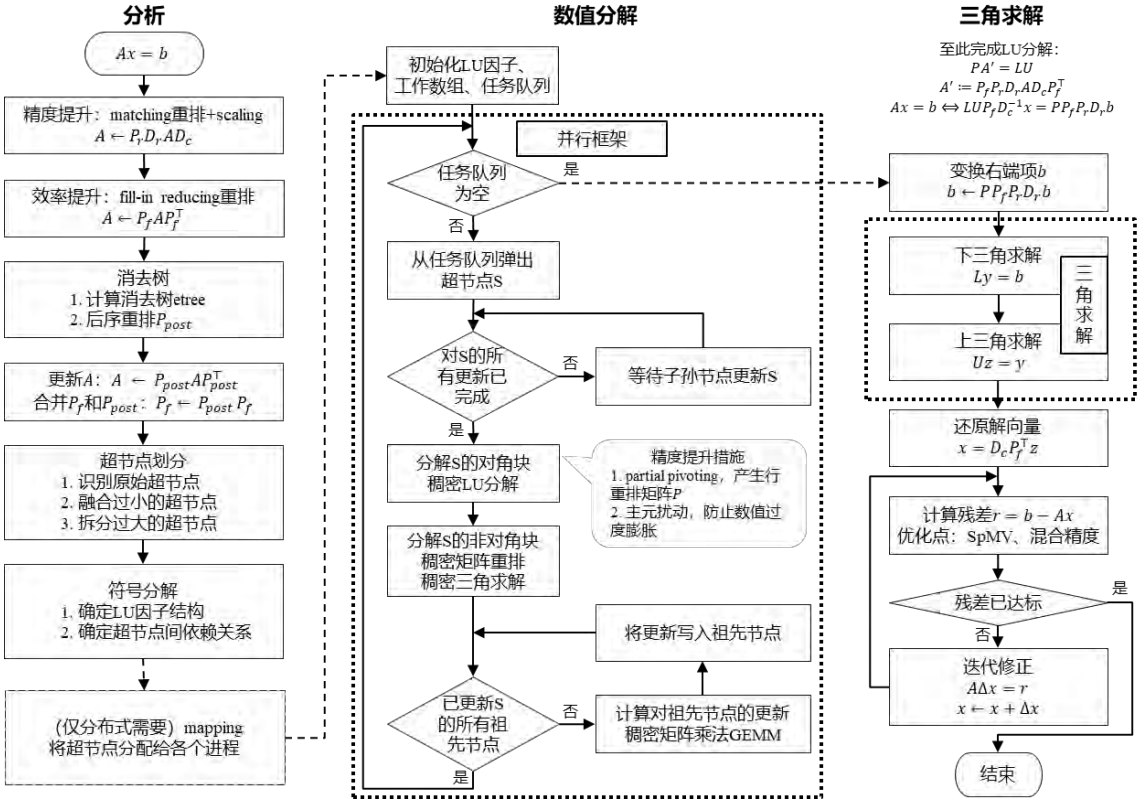

图2.34 超节点直接法算法流程

为支撑映射与调度决策，求解器建立了基于浮点运算量的性能建模机制，在分析阶段估计各超节点的计算规模，并结合分阶段、分工作负载的拟合模型，对不同超节点和执行阶段的运行代价进行量化刻画。该模型不追求精确预测单次运行时间，而是为映射、负载分布以及通信优化提供统一、可解释的成本度量。

总体而言，该并行超节点直接法通过将结构分析、映射、通信组织与性能建模有机结合，实现了一种结构驱动、静态为主的并行执行模式。在保证数值正确性与稳定性的前提下，该方法在负载均衡、通信开销控制以及并行可扩展性之间取得了良好的工程折中，适用于大规模稀疏线性系统的高性能求解。

（八）迭代法：自适应初值选取

在迭代法求解大规模稀疏线性系统时，初始猜测的质量对收敛效率具有显著

影响。实践表明，选择一个合适的初始解可以大幅减少达到预定精度所需的迭代次数，甚至决定迭代过程能否顺利收敛。相反，若初始猜测与真实解相差较大，迭代算法往往需要经历较长的无效搜索阶段，增加不必要的计算开销。

合理的初始猜测不仅有助于加快收敛，还能够提升迭代过程的稳定性。在缺乏有效先验信息的情况下，迭代算法可能在误差空间中沿着不利方向反复迭代，表现为收敛缓慢、振荡，甚至出现发散风险。尤其是在病态矩阵或强非均匀谱分布的情形下，初始误差过大可能导致算法在错误的误差分量上消耗过多迭代步数，从而显著降低整体求解效率。

从误差传播的角度看，设第 k 次迭代的误差为

$$e^{k+1} = Ge^k$$

其中 G 为迭代算子。初始 e^0 将直接影响误差衰减的速度。当初始猜测在某些“慢衰减模态”（如低频误差或与小特征值相关的方向）上分量较大时，即使迭代算法本身具有良好的渐近收敛性质，也可能在前期表现出明显的收敛迟滞，甚至出现振荡或数值不稳定现象。该问题在病态矩阵或谱分布高度不均匀的场景下尤为突出。

在缺乏显式先验解的情况下，可在正式Krylov迭代之前引入低成本的预迭代过程，例如执行少量Jacobi、Gauss-Seidel或弱预处理迭代，如图2.35所示，用以快速削弱主要误差分量[75]，构造一个具有一定精度的近似解 $x^0 = \tilde{x}$，而非传统的 0 初始解 $x^0 = 0$。通过构造初始解，可以使得初始残差满足：

$$\left||b - A\tilde{x}|\right| \ll ||b||$$

该过程计算代价较低，却能够显著改善主迭代阶段的收敛表现。

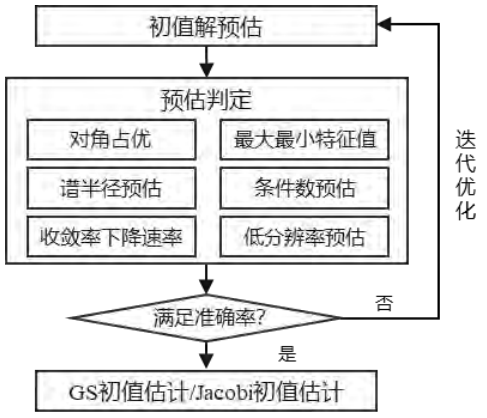

图2.35 自适应初值选取算法流程

总体而言，自适应地构造初始猜测是一种低成本、高收益的关键流程优化手段。

该方法不改变线性系统本身的数学性质，也不依赖特定的迭代算法实现，却能够显著缩短收敛路径、减少迭代次数，并提高迭代法在复杂工程问题中的整体鲁棒性与效率。在迭代法求解大规模稀疏线性系统时，初始猜测 $x^{(0)}$ 的选取对收敛速度具有重要影响。数值实践表明，合适的初始解可以显著减少达到预设精度所需的迭代步数，甚至决定迭代过程是否收敛。反之，若初始误差

（九）**迭代法：对角缩放**

在迭代法求解大规模稀疏线性系统的过程中，对角缩放（Diagonal Scaling）是一种简单但极其有效的预处理手段，其主要目的在于改善线性系统的数值尺度与条件性质，从而提升迭代算法的收敛速度与数值稳定性。

在实际工程问题中，线性系统往往来源于多物理场耦合或复杂离散模型，不同未知量及其对应方程可能具有显著不同的量纲和数值尺度。例如，位移、压力、电势等变量在同一系统中共存时，其数值范围可能相差数个数量级。若直接对原始矩阵进行迭代求解，容易导致残差向量中不同分量的幅值差异过大，使得迭代过程在某些方向上收敛缓慢，甚至诱发数值不稳定或舍入误差放大问题。

对角缩放的基本思想如图2.36所示，通过引入合适的对角矩阵，对原始线性系统进行等价变换，使矩阵在数值尺度上更加均衡。典型形式为：

$$D_L A D_R y = D_L b$$

$$x = D_R y$$

其中 D_L、D_R 为对角矩阵。上式变换前后在数学上等价，不改变原问题的精确解。

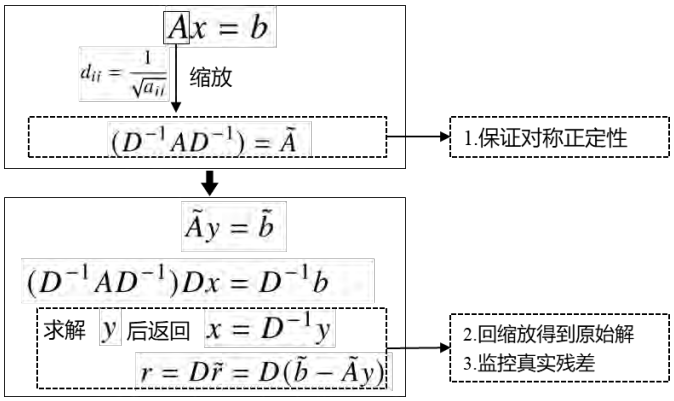

图2.36 对角缩放示意图

假设考虑对称双边缩放：

$$D^{-1}AD^{-1}y = D^{-1}b$$

$$x = D^{-1}y$$

其中 $D = diag\{\sqrt{|a_{11}|}, \sqrt{|a_{22}|}, \dots, \sqrt{|a_{nn}|}\}$ 通常取为矩阵对角元的绝对值的开根。通过上述缩放操作，可以显著降低矩阵行或列之间的尺度差异，使得迭代过程中误差分量的衰减更加均匀。对于依赖内积、范数或残差比较的迭代方法（如CG、GMRES、BiCGSTAB等）而言，这一点尤为重要，因为缩放后的系统能够减少数值主导方向的不平衡，提升Krylov子空间构造的有效性。从谱性质角度看，对角缩放通常不会改变矩阵的非零结构，但可以改善特征值的分布特性，降低条件数：

$$\kappa(D^{-1}AD^{-1}) \ll \kappa(A)$$

从而减少迭代过程中对参数调节（如收敛容限、重启策略等）的敏感性。

在整体求解流程中，对角缩放通常作为迭代法启动前的轻量级预处理步骤，与自适应初值构造、区域分解或其他预处理技术自然衔接。一方面，对角缩放能够显著降低初始残 $r^0 = b - Ax^0$ 在不同分量上的不平衡程度，使初始猜测在数值上更加“友好”；另一方面，其计算代价极低，却往往能带来可观的收敛性能提升，是工程实践中性价比极高的基础优化手段。

（十）迭代法：区域分解

在大规模稀疏线性系统的迭代求解过程中，区域分解是一类重要的结构性预处理技术，其核心目标是通过将原始计算区域或未知量集合划分为若干相对独立的子区域，在子区域层面构造局部求解器，从而改善整体迭代过程的并行性、数值稳定性与收敛效率。

在实际工程问题中，离散后的线性系统往往来源于偏微分方程（PDE）的空间离散，不同区域之间的耦合强度、物理特性以及网格分辨率可能存在显著差异。若直接对整体系统进行迭代求解，容易出现局部误差传播缓慢、全局低频误差衰减效率低等问题，进而限制迭代法在大规模并行环境下的性能表现。区域分解方法正是通过显式引入子区域层级的求解结构，来缓解上述问题。

考虑线性系统

$$Ax = b, \quad A \in \mathbb{R}^{n \times n}$$

将未知量集合划分为N个子区域（或子空间）：

$$\{1,2,\dots,n\} = \bigcup_{n=1}^{m}(\mathcal{L}_i)$$

对应地，可将矩阵 A 在代数意义上划分为若干局部子矩阵：

$$A_i = R_i A R_i^T, \qquad i = 1,\dots,N$$

其中 R_i 为限制算子（Restriction Operator），用于从全局向量中提取子区域 $\mathcal{L}_i$ 上的分量；A_i 表示第 i 个子区域上的局部算子。区域分解方法的基本思想是：通过对子区域局部问题的求解或近似求解，来构造全局系统的预处理器：

$$A_i x_i = b_i$$

在迭代法中，最常见的区域分解形式为加性Schwarz方法（Additive Schwarz Method，ASM），其预处理子可写为：

$$M^{-1} = \sum_{i}^{N} R_i^T A_i^{-1} R_i$$

在Krylov子空间方法（如CG、GMRES、BiCGSTAB）中，对应的预处理系统为

$$M^{-1}Ax = M^{-1}b$$

其中每个 A_i^{-1} 通常并不需要精确求解，而是采用局部直接法、ILU或少量迭代作为近似；

子区域求解彼此独立，天然适合并行实现。

根据子区域之间是否存在自由度重叠，区域分解方法可分为（1）非重叠区域分解：如Block Jacobi、FETI和BDDC等；（2）和重叠区域分解，强子区域之间的信息传递能力，通常可获得更好的收敛性质。以Block Jacobi为例构造无重叠预处理子：

$$M_{BJ}^{-1} = \sum_{i}^{N} R_i^T A_{ii}^{-1} R_i$$

其中 A_{ii} 是矩阵 A 的对角块。Block Jacobi方法可以视为一种无重叠、加性Schwarz 型的区域分解方法，其通过对角块的独立求解构造预处理器，具有实现简单、并行性强的优点。

（十一）迭代法：基于子区域迭代的混合精度算法

在大规模稀疏线性系统的迭代求解中，计算效率与数值精度之间存在天然矛

盾：低精度计算可显著提升吞吐与能效，但误差累积会削弱收敛性甚至导致发散。为在保证收敛可靠性的前提下充分利用低精度算力，本求解流程引入基于子区域迭代的粗–精度协同算法，并采用选择性混合精度（Selective Mixed Precision）、迭代求精（Iterative Refinement / Residual Correction）和稳定更新（Reliable Updates）的机制相结合的设计。该方法的核心思想是：以子区域（子域）迭代为基本框架，将全局求解过程拆分为“局部迭代更新和全局一致性校正”的结构。在此基础上，对不同计算步骤按其对收敛性的敏感程度进行精度分配——即“关键步骤用高精度，主耗时算子用低精度”。具体而言，针对数值稳定性影响更大的环节（例如预处理器的构建与更新、残差的计算与收敛判据评估等），采用FP64/FP32以保证稳定性；而对计算占比极高、且更适合向量化/张量化加速的矩阵向量乘积环节，则采用 FP16 进行加速，以提升整体迭代吞吐。算法步骤如下：

除了针对区域的混合精度，如何对混合精度后的求解做修正也对迭代法的收敛性之起到了至关重要的因素，否则将会出现由于精度下降导致迭代步数大幅增加，致使求解总时长反而增加的情况。在混合精度的设置下，第k次迭代的残差更新可以表示为：

$$\tilde{r}^k = b - \tilde{A}\tilde{x}^k$$

其中 $\tilde{A},\ \tilde{x}^k$ 表示在低精度环境下参与SpMV的近似量。由于低精度计算引入的舍入误差 $\tilde{r}^k$ 与真实残差 $r^k = b - Ax^k$ 之间不可避免地产生偏差，该偏差如果长期积累，将会影响迭代方向的有效性。为抑制上述误差累积并维持整体收敛性，必须对残差实行更新和修正。典型的做法是在每隔一定的迭代步数后，采用高精度重新计算残差 r^k，并基于该高精度残差重新构造解的修正量 δx^k，从而更新当前解：

$$\delta x^k + x^k \to x^{k+1}$$

下面基于 CG 算法，介绍采用稳定更新策略使用混合精度的思路：

1) 计算 $r^0 = b - Ax^0$记为初始残差；

2) 将初始残差转移到低精度 $r^0 \to \tilde{r}^0$，设定初始低精度解 $\tilde{x}^0$；

3) 计算低精度残差并判断是否需要更新：$\left||\tilde{r}^k|\right| > \epsilon$；

4) 如果低精度残差范数大于阈值，那么执行残差修正策略；

5) 最终在低精度的运算效率下，对残差进行可靠更新，获得高精度的收敛解。

总体而言，该基于子区域迭代的粗－精度协同方案在工程上实现了三方面收益：其一，通过在主要计算瓶颈（SpMV）上采用低精度获得显著加速；其二，通过对关键步骤保留高精度计算保障收敛鲁棒性；其三，通过迭代求精的周期性纠偏机制控制误差累积，使混合精度求解在复杂工程矩阵上具备可控、可预测的数值行为。该策略与现有预处理器和主迭代法具有良好的兼容性，可作为迭代求解流程中的通用加速与稳定性增强手段。

2.2.1.5 线性方程组求解器功能特性

线性方程组求解器作为用户求解大规模线性代数方程组的重要计算工具，应提供矩阵数据输入、求解算法配置、求解执行以及结果获取等基础功能。针对不同规模和不同性质的线性系统，求解器应能够支持多种求解策略，包括基于稀疏矩阵分解的直接求解方法以及基 Krylov子空间方法的迭代求解算法，并能够结合多种预处理技术以提高求解效率。主流线性方程组求解器应支持如下所述功能。

（一）问题类型支持

求解器应支持多种矩阵类型，包括实数或复数矩阵、对称矩阵、Hermitian矩阵、正定矩阵以及一般非对称矩阵，并可根据矩阵结构自动选择合适的分解算法，例如Cholesky分解、LDL^T分解或LU分解。求解器适用于来自有限元、有限体积、有限差分等数值离散方法产生的大规模稀疏线性系统，也支持多右端问题的高效求解。

（二）求解能力

应提供多种求解策略，以适应不同规模与不同性质的线性系统。对于中等规模问题或需要高数值稳定性的场景，求解器应支持基于稀疏矩阵分解的直接求解方法，例如LU、Cholesky 或 LDL^T分解。对于大规模问题，求解器应支持基于Krylov子空间方法的迭代求解算法，例如CG、GMRES、BiCGStab等方法，并能够结合多种预处理技术提高收敛速度。预处理器应支持多种策略，例如不完全分解（ILU/ICC）、块预处理以及代数多重网格（AMG）等。

（三）数据输入

应支持多种标准矩阵输入格式，其中最常用的是Matrix Market（MTX）格式。MTX是一种广泛使用的稀疏矩阵文本格式，用于描述矩阵的规模、非零元数量以及每个非零元素对应的行列索引和值信息。该格式具有良好的可读性和通用性，被

广泛用于数值计算软件以及公开矩阵测试集（例如 SuiteSparse Matrix Collection）中。求解器应能够直接读取MTX格式的矩阵数据，并自动构建内部稀疏矩阵存储结构以用于后续计算。除文本格式外，求解器还应支持MTB（二进制矩阵）格式，该格式采用二进制方式存储矩阵结构及数值信息，相比文本格式能够显著减少文件体积并提高数据读取效率，其设计方式类似于PETSc等高性能计算软件中常用的二进制矩阵存储格式。此外，当用户未提供右端向量时，求解器应能够自动生成默认右端向量，例如使用全1向量作为右端并计算对应解向量，用于验证矩阵分解和求解流程的正确性。此外，求解器还应支持多右端向量输入，以满足批量求解或多载荷计算场景下的需求。

（四）**日志和结果输出**

应提供完整的日志输出与求解过程信息记录机制。在求解开始阶段，系统应能够输出输入矩阵的基本信息，例如矩阵规模、非零元数量、矩阵类型以及结构特征等。运行过程中，日志应记录当前使用的计算资源信息，包括MPI进程数量、OpenMP线程数量以及计算节点配置等。对于迭代求解算法，日志应能够输出每一步迭代的残差范数以及迭代次数等信息；对于直接求解方法，系统应能够输出矩阵分析、数值分解以及回代求解等阶段的性能统计。在求解结束后，系统应输出最终残差、收敛状态以及总体计算时间等信息。

（五）**参数设置**

应能够设置求解算法类型，例如选择直接法或迭代法，并能够配置具体算法参数，例如Krylov子空间维度、最大迭代次数以及残差收敛阈值等。对于迭代求解方法，系统应允许用户选择预处理策略并配置相关参数，例如ILU分解级别或多重网格层数。同时，求解器应允许配置矩阵重排序策略、缩放策略以及并行计算相关参数。

（六）**功能接口**

应能够通过接口加载矩阵与右端向量数据，并通过参数配置接口选择求解算法和预处理策略。求解器应提供统一的求解流程接口，接口设计应支持重复求解场景，例如在矩阵不变但右端向量变化的情况下复用已有计算结果。

（七）**并行加速**

应支持基于OpenMP的多线程并行和基于MPI的分布式并行，部分情况下还应

支持MPI与OpenMP的混合并行模式。

2.2.2 非线性方程组求解器技术

2.2.2.1 非线性方程组求解器简介

在物理仿真与数值计算领域，非线性问题占据了核心地位，其本质反映了物理世界中响应与激励的非比例特性。无论是结构力学中的大变形几何非线性与接触碰撞，流体力学中的纳维-斯托克斯方程对流项引起的湍流耦合，还是多物理场模拟中材料属性随状态变量剧烈变化的本构关系，归根结底均需转化为形式为$F(x) = 0$的大规模非线性方程组[76]。与线性问题不同，此类方程组通常不存在封闭的解析解，且解的唯一性与存在性难以保证，因此数值求解的核心挑战在于如何在缺乏全局信息的情况下，通过局部线性化迭代，高效、稳定地逼近真实物理解。

2.2.2.2 非线性方程组求解器软件架构

非线性方程组求解器的整体架构图如图2.37所示。

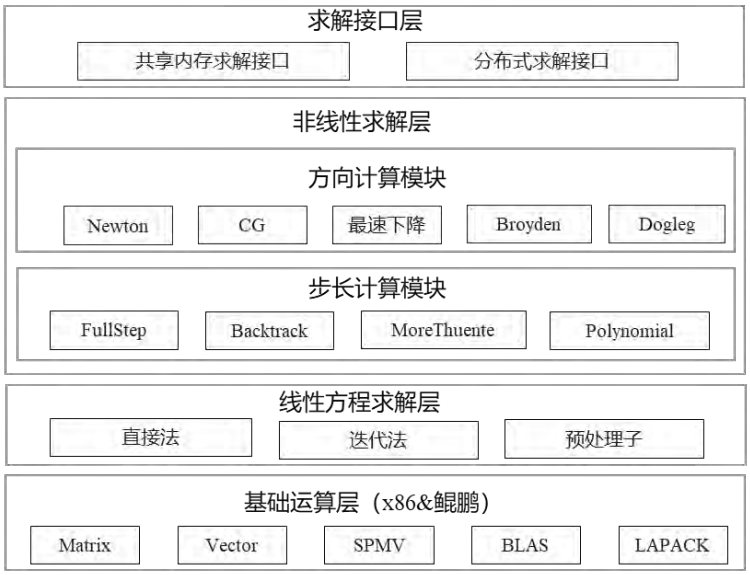

图2.37 非线性方程组求解器软件架构

可供用户直接调用的是求解接口层，为了适配现代计算集群，该层集成了共享内存接口与分布式求解接口，前者利用多核并行优化局部计算，后者则通过跨节点通信支持超大规模问题的并行处理。

非线性求解层是核心求解层，采用经典的“方向-步长”分离设计模式，将复杂的非线性迭代过程解耦为两个子模块：

1) 方向计算模块：负责计算迭代更新的方向。该模块具有可拓展性，封装了包括Newton法、非精确牛顿法、Broyden拟牛顿法、CG及最速下降法等多种策略，以适应不同物理问题的数学特性。

2) 步长计算模块：负责控制解的更新幅度以保证全局收敛。该模块通过统一的接口管理各类线搜索技术，包括FullStep、Backtrack、Polynomial以及MoreThuente算法，确保非线性迭代的稳定性。

非线性求解过程中通过线性化，最终会转变为线性方程求解，因此非线性求解器底层依赖线性求解层。根据矩阵规模与性质的不同，系统支持基于超节点方法的直接法，以获取极高的数值稳定性；同时提供丰富的迭代法与预处理子组合，如利用代数多重网格或不完全分解技术，显著改善线性系统的条件数，从而加速整个非线性迭代过程。通过这种从全局策略到底层计算的垂直调优，本求解器能够灵活应对各类非线性问题，完成高效求解。

2.2.2.3 非线性方程组求解器整体求解流程

非线性方程组求解器的核心求解机制是一个由初始猜测值出发，通过不断迭代逼近真实解的流程。对于目标方程组 $F(x) = 0$，整个求解工作流可分为如图2.38所示的四个紧密衔接的关键环节。

（一）线性化阶段

首先是基于泰勒展开的局部线性化阶段。由于缺乏直接求解多维非线性系统的解析手段，求解器会在当前的迭代状态点 x_k 处对非线性算子进行一阶泰勒展开，将其近似表达为 $F(x_k + p) = F(x_k) + J(x_k)p$，其中 $J(x_k)$ 代表系统当前的雅可比矩阵。通过令该线性近似式等于零，将原本的非线性问题，转化为了在一个局部邻域内求解线性方程组的子问题[26]。

（二）方向计算

完成线性化构造后，流程进入到方向计算阶段。该阶段的核心计算目标是求解上述的线性系统 $J(x_k)p_k = -F(x_k)$，从而获取向真实解逼近的搜索方向 p_k。在这一过程中，求解器根据问题的物理规模与雅可比矩阵的数学性质调用不同的底层

算法。例如，常规问题会直接依靠精确的牛顿方向；超大规模问题会启用非精确牛顿法结合迭代法来高效获取近似方向；而在雅可比矩阵难获取的工况下，则会引入Broyden等拟牛顿法。

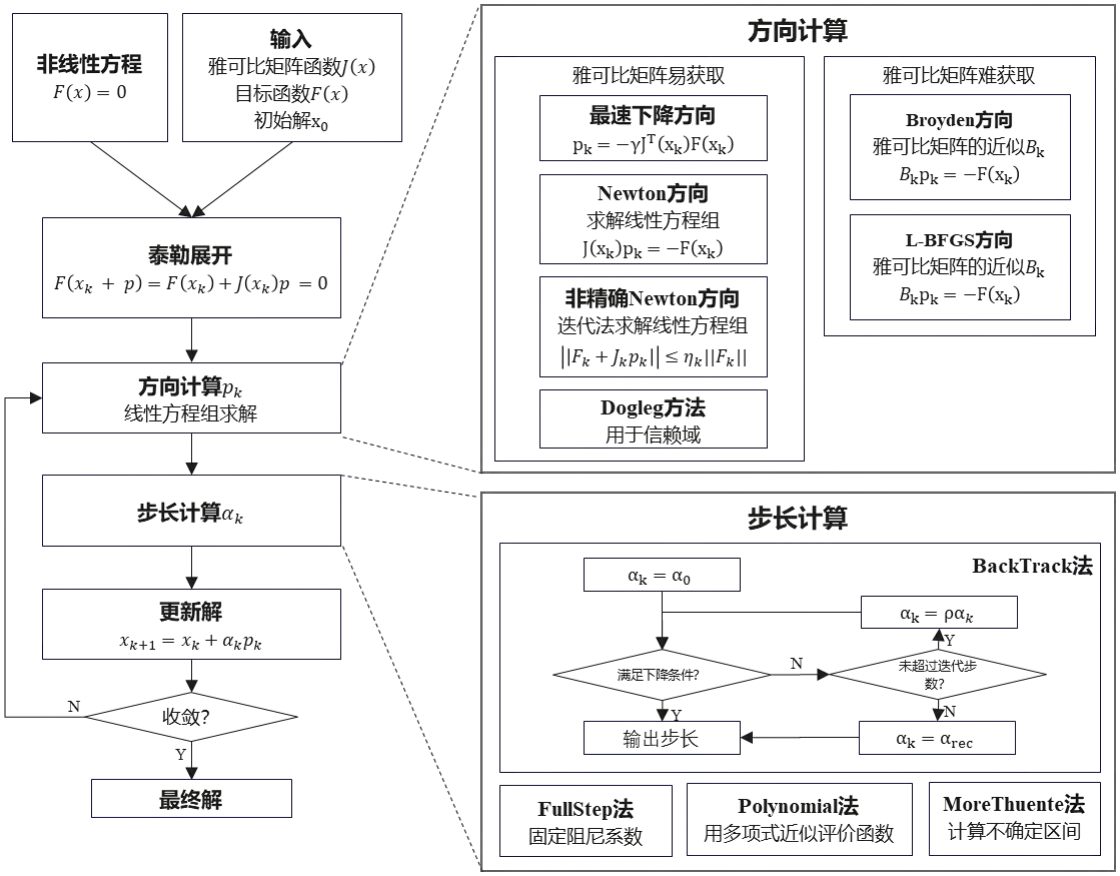

图2.38 非线性方程组求解流程

（三）步长计算

在确立了搜索方向后，直接沿该方向前进全步长往往面临越过真实解或导致残差震荡的风险，因此流程会紧接着切入步长计算环节。求解器此时会引入一个标量步长因子 α_k，求解器可以选择不同的步长计算策略：若处于良好的二次收敛域内，则直接施加全步长以追求极致效率；若处于复杂的非线性阶段，则会使用Backtrack回溯、Polynomial多项式插值或 MoreThuente等策略[77]，以动态调控步幅，确保目标残差获得充分的下降。

（四）更新解

经过方向规划与步长计算，流程最终来到更新解与收敛判定阶段。求解器利用位移更新公式 $x_{k+1}=x_k+\alpha_k p_k$ 将物理系统的当前状态推进到下一个迭代点。随后，求解器会评估新状态下的残差范数以及解的相对变化率。若尚未满足收敛条件，

则系统会将最新的状态 x_{k+1} 作为新的起点迭代计算，直至求解成功。

2.2.2.4 非线性方程组求解器关键技术

非线性方程求解器基于主流的Newton法及其变体，核心算法围绕方向计算和步长计算两大维度展开，并通过底层的线性求解技术支撑，最终实现迭代收敛。

在方向计算的关键技术上，求解器提供了多层次的算法选择以应对不同的物理特性。基础的Newton法利用精确的Jacobian矩阵信息，在解的邻域内能够实现极快的二次收敛，是处理常规非线性问题的首选。针对计算资源受限或Jacobian矩阵难以显式构建的复杂方程，Broyden拟牛顿法通过迭代历史数据来近似Jacobian矩阵的逆，避免了昂贵的导数计算。对于超大规模的三维物理场问题，非精确牛顿法被设计用来动态调整内层线性方程组的求解精度，即在非线性迭代初期允许较大的线性残差，从而大幅节省计算时间。此外，针对雅可比矩阵非正定或初始猜测较差的场景，Dogleg算法作为一种先进的信赖域策略，巧妙地结合了最速下降法的全局稳定性和牛顿法的局部快速收敛性，通过计算柯西点与牛顿点之间的折线路径，自动调节更新方向，确保算法在较差工况下依然稳健[26]。而CG与最速下降法则作为低内存消耗的备选方案，常用于辅助解决大规模问题。

在步长计算技术方面，求解器通过精细的线搜索策略来实现全局收敛。FullStep全步长策略主要用于非线性较弱或接近收敛的阶段，以最大化迭代效率。当全步长导致残差震荡或发散时，Backtrack回溯法会基于简单的比例缩减机制快速寻找下降点。为了获得更优的下降量，Polynomial多项式插值法利用当前点的函数值与导数信息构建二次或三次模型，通过解析模型的极值点来预测最优步长。对于对收敛条件极为敏感的强刚性问题，MoreThuente算法提供了更为严格的保障，它在搜索过程中强制满足Wolfe条件，即同时满足充分下降条件和曲率条件[77]，有效避免了步长过小导致的伪收敛或步长过大引起的数值不稳定，是处理高难度非线性物理场的关键技术。

上述非线性迭代过程的效率，最终取决于底层线性方程求解层的性能。针对刚性矩阵，求解器集成了基于超节点（Supernodal）技术的直接法，利用分块计算极大提升了缓存命中率和数值稳定性。而针对超大规模的稀疏系统，则主要依赖迭代法配合先进的预处理技术，其中代数多重网格（AMG）能有效消除低频误差，而

不完全分解（ILU）等预条件子则能改善矩阵的条件数，两者显著加速了非线性迭代中每一线性子步的求解速度。

2.2.2.5 非线性方程组求解器功能特性

非线性方程组求解器作为用户求解非线性代数系统的重要计算工具，应提供初始解设置、求解算法配置、非线性迭代执行以及结果获取等基础功能。针对复杂非线性问题的求解需求，求解器应能够支持多种非线性迭代策略，并通过方向计算与步长控制相结合的算法框架实现稳定高效的求解过程，同时能够调用底层线性求解模块完成非线性迭代过程中产生的线性子问题。主流非线性方程组求解器应支持如下所述功能。

（一）问题类型支持

求解器应适用于由非线性偏微分方程离散化得到的大规模代数系统，例如流体动力学、结构力学、电磁场以及多物理场耦合问题中的非线性系统。求解器应能够处理大规模稀疏非线性问题，并应支持用户提供残差函数以及可选的雅可比矩阵信息，从而适应不同物理模型和数值离散方法产生的非线性系统。

（二）求解能力

方向计算模块应负责计算非线性迭代更新方向，并应支持多种策略，例如Newton方法、非精确Newton方法、Broyden拟牛顿方法以及基于梯度信息的迭代方法（如共轭梯度法或最速下降法）。步长控制模块应负责控制每一步迭代的更新幅度，并应支持多种线搜索算法，例如 FullStep、Backtracking、Polynomial线搜索以及More-Thuente线搜索策略。

（三）数据输入

应支持用户提供初始解向量以及非线性残差函数接口，并应允许用户提供雅可比矩阵或其计算方法。在需要显式矩阵表示的情况下，雅可比矩阵应支持标准稀疏矩阵输入格式。对于规模较大的问题，求解器还应支持矩阵-向量乘算子形式的输入，从而实现矩阵无显式存储（matrix-free）的求解模式，以减少存储开销并提高计算效率。

（四）日志和结果输出

应提供完整的求解过程日志输出，以便用户分析算法收敛行为和计算性能。日

志信息应包括初始残差大小、每一步非线性迭代的残差范数、步长信息以及方向计算过程中线性子问题的求解状态等。在求解完成后，系统应能够输出最终残差、非线性迭代次数以及收敛状态，并应提供求解时间统计与性能分析信息。

（五）**参数设置**

应提供灵活的参数配置机制，以便用户根据具体问题特性调整求解策略。用户应能够设置最大非线性迭代次数、残差收敛阈值、线搜索策略以及步长控制参数等。同时，求解器应允许用户选择不同的方向计算方法，例如Newton方法或拟牛顿方法，并应支持配置内部线性求解器及其预处理策略。

（六）**功能接口**

应支持通过接口提供初始解向量、残差函数以及可选的雅可比矩阵计算函数，并通过统一接口执行非线性求解过程。接口设计应支持模块化调用，并能够在迭代过程中获取当前解向量、残差信息以及收敛状态。

（七）**并行加速**

应支持基于OpenMP的多线程并行和基于MPI的分布式并行，部分情况下还应支持MPI与OpenMP的混合并行模式。

2.2.3 特征值求解器技术

2.2.3.1 特征值求解器简介

特征值计算是数值计算求解中一个重要的方向。在各类仿真场景中，尤其是结构、电磁仿真场景，模态分析、屈曲分析等任务最终会转化为求解大型稀疏矩阵的特征值问题。针对大型稀疏矩阵研究高效稳定的特征值求解算法，在CAE仿真领域的实际应用中具有十分重要的意义。另外特征值算法还需要支持各种特性，如求解多个最小、最大特征值，求解指定区间内的特征值，求解高阶特征值，处理特征值重根等问题。

2.2.3.2 特征值求解器软件架构

本求解器采用分层模块化设计，旨在解决大规模稀疏矩阵及复杂工程应用中的特征值问题，特征值求解器的整体架构图如下：

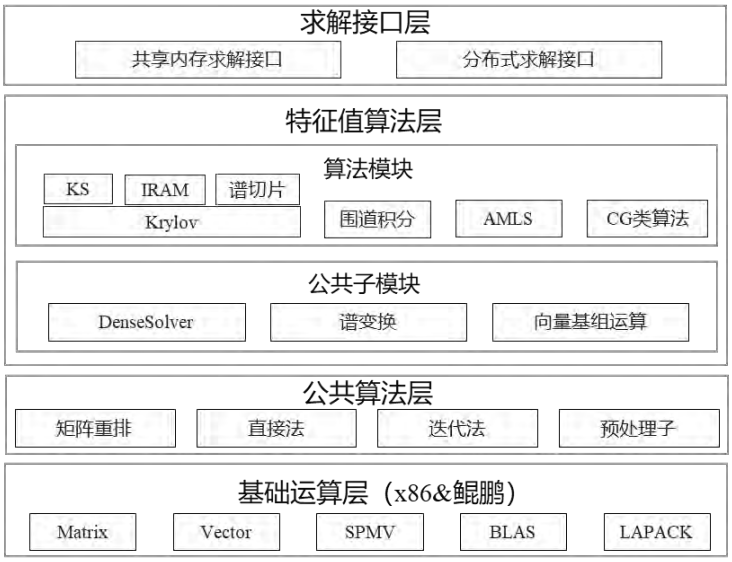

图2.39 特征值求解器软件架构

特征值求解器的最外层是用户调用接口层，为了兼顾不同计算场景的性能需求，该层提供了共享内存并行接口与分布式并行接口。前者主要针对单机多核环境，通过细粒度的线程级并行优化，最大化利用本地内存带宽；后者则面向大规模高性能计算集群，支撑起超大规模矩阵的数值计算任务。

核心算法层作为特征值求解器的大脑，集成了多种针对不同物理问题的先进求解策略。Krylov子空间系列算法通过将大系统投影到低维空间，能够稳健地捕获稀疏矩阵的端部特征值[78]；针对区间特征值求解问题，谱切片技术为寻找指定区间内的所有特征值提供了求解路径[79]，围道积分算法则利用复平面上的数值积分实现特征空间的正交投影，具有天然的并行优势[80]。对于高阶特征值问题，AMLS通过多级降阶策略显著提升了计算效率[81]。此外，CG类算法则为对称正定系统提供了高效的梯度搜索手段[82]。

在算法层的底层，三大子模块共同构成了特征值求解器的基座。DenseSolver模块专注于处理经由降阶后产生的稠密小矩阵特征值问题，通过深度优化的数学库确保了投影空间的计算精度与效率。谱变换模块利用谱位移与求逆等数学手段，将用户关注的内部特征值移动至频谱边缘，从而极大地加速了Krylov方法的收敛速度，并能灵活转换特征值问题的表现形式。最后，所有的空间构建任务都基于向量基组运算模块，它负责执行复杂的子空间正交化、基向量扩展以及基组压缩等操作，确保了在迭代过程中子空间始终保持优良的数值稳定性，为整个求解器的健壮性

提供了底层保障。

2.2.3.3 特征值求解器整体求解流程

总结来说，特征值求解器的核心思想是通过投影方法将大规模原始问题降阶为小规模问题进行近似求解。虽然不同算法的原理不同，但其通用的关键求解流程可归纳如图2.40。

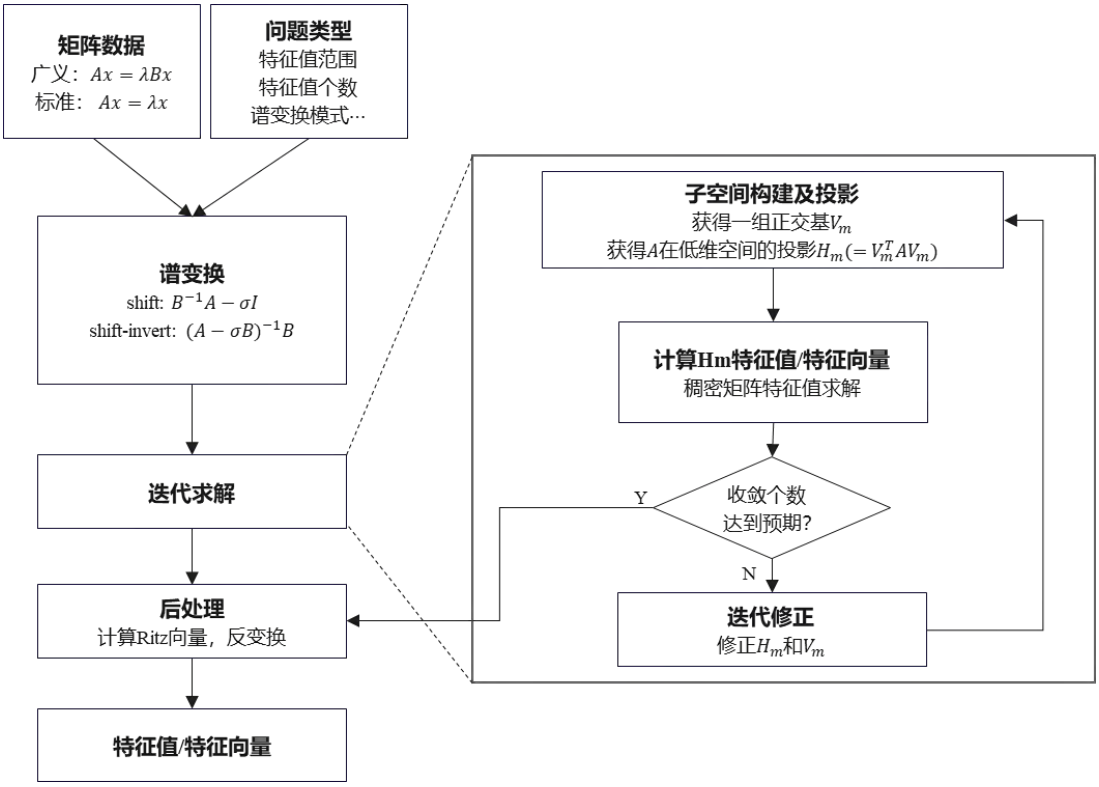

图2.40 特征值求解流程

（一）谱变换与预处理

该步骤旨在通过数学手段改善问题的求解性质。通常利用谱位移或移位反转等操作，将用户关注的目标特征值（如中间谱）映射至频谱边缘，以此改善算子的谱分布特性，从而加快迭代收敛；对于梯度优化类算法，则在此阶段构建高效的预条件子，以显著加速梯度搜索的收敛过程。

（二）子空间构建

这是不同算法策略产生区别的核心环节，其目的是构建一个包含解信息的低维空间。Krylov子空间类算法主要通过连续的矩阵向量乘积生成正交基底；AMLS类算法则基于多级子结构树，生成组件模态来构建基底[81]；而围道积分法则通过

在复平面上的数值积分操作来生成投影矩阵[80]。

（三）投影与小规模问题构建

在此步骤中，求解器利用瑞利-里兹等投影方式执行模型降阶。它将原始的大规模稀疏矩阵投影到前序步骤构建的低维子空间中，从而生成一个能够代表原系统特征的稠密小规模矩阵。

（四）稠密矩阵求解

针对降阶后产生的稠密小矩阵，通常采用QR算法或分治法等求解技术，快速、精确地计算出投影后的特征对，即获取Ritz值与Ritz向量，作为原问题的近似解。

（五）收敛性检测与迭代修正

流程的最后一步是计算残差范数以评估近似解的精度。对于Krylov等迭代方法，若未达到精度要求，则通过重启技术剔除无效方向并压缩子空间，进入下一轮迭代循环。

2.2.3.4 特征值求解器关键技术

稀疏矩阵特征值求解算法普遍基于Krylov子空间的方法，用于求解部分特征值及特征向量。隐式重启Arnoldi算法和Krylov-Schur算法是业界公认的的高效算法，通过结合隐式QR算法提高基于重启策略的Krylov子空间算法的收敛速度[78]，被广泛应用于各个领域，可同时处理对称、非对称特征值问题。

此外，某些领域的仿真对特征值求解器有两个额外的特性需求，即求解指定区间内全部特征值，以及求解高阶特征值（成百上千阶特征值）。区间特征值求解主要用到两种高效算法，即围道积分法和谱切片法。这些算法能够利用特征值的分布性质进行快速筛选[79]，从而避免了对所有特征值的遍历。当矩阵规模庞大或结构复杂时，特征值的计算数量巨大，不仅耗时较长，而且容易导致计算不稳定，结果精度难以保证。针对这类高阶特征值求解难题，主要使用两类高效算法，即多级子结构法（如AMLS）和共轭梯度算法（如LOBPCG、GCG）。这些算法具备以下特点：首先，能够准确捕捉高阶特征值，即使在特征值分布广泛且密集的情况下也能保持高精度；其次，算法的计算效率高，能够处理大规模矩阵的求解问题；最后，算法具备数值稳定性，以减少计算过程中的误差积累，确保结果的准确性。

2.2.3.5 特征值求解器关键技术

特征值求解器作为用户求解矩阵特征值与特征向量的重要计算工具，应提供矩阵数据输入、算法配置、特征值计算以及结果获取等基础功能。针对大规模稀疏矩阵特征值问题，求解器应能够支持多种基于Krylov子空间的特征值计算算法，并能够通过谱变换技术提高特定谱区间特征值的计算效率，同时能够调用底层线性求解模块以完成相关线性系统求解。主流特征值求解器应支持如下所述功能。

（一）问题类型支持

求解器应能够处理实数或复数矩阵，并应支持对称、Hermitian以及一般非对称矩阵类型。该求解能力应适用于结构振动分析、电磁场计算、稳定性分析以及其他科学计算和工程仿真中的特征值问题。系统应支持计算矩阵的部分特征值及对应特征向量，例如最大特征值、最小特征值或指定谱区间内的特征值。

（二）求解能力

应支持多种基于Krylov子空间的特征值计算方法，还应支持谱变换技术，例如shift-and-invert方法，以提高对特定谱区间特征值的计算效率。在谱变换模式下，系统应能够调用底层线性求解器完成相关线性系统求解。

（三）数据输入

应支持输入一个或两个稀疏矩阵作为计算对象。矩阵输入应支持标准稀疏矩阵格式，例如 Matrix Market（MTX）文本格式以及二进制矩阵格式（MTB）。此外，求解器还应允许用户指定需要计算的特征值数量、目标谱区间或目标特征值类型（例如最大模特征值或最小特征值）。

（四）日志和结果输出

应提供完整的迭代过程日志输出，用于记录特征值计算过程中的关键数值信息。日志信息应包括当前迭代步数、Krylov子空间维度、已收敛特征值数量以及对应的残差范数等。在求解完成后，系统应能够输出所有收敛特征值及其对应特征向量的残差，并提供整体求解时间和计算性能统计信息。

（五）参数设置

应提供灵活的参数配置机制，使用户能够根据问题规模和矩阵特性调整算法行为。用户应能够设置需要计算的特征值数量、Krylov子空间维度、最大迭代次数

以及收敛阈值等算法参数。同时，求解器应允许用户配置谱变换策略及其相关参数，例如 shift 值以及内部线性求解器的配置参数。

（六）**功能接口**

应支持通过接口加载矩阵数据并配置相关算法参数，然后调用求解接口执行特征值计算。接口应能够返回特征值和特征向量计算结果，并允许用户按需获取部分结果或全部结果。同时，接口设计应支持在求解过程中获取当前收敛状态和中间计算信息。

（七）**并行加速**

应支持基于OpenMP的多线程并行、基于MPI的分布式并行，还应支持MPI与OpenMP的混合并行模式。

2.3 AI 辅助求解技术

近年来，以大语言模型为代表的人工智能技术爆发式突破，为突破求解器的性能瓶颈提供了全新的技术范式。这一新兴领域常被称为学习优化（Learning to Optimize，L2O）、AI辅助优化（AI for Optimization，AI4OPT）或机器学习辅助优化（Machine Learning for Optimization，ML4OPT）。

AI辅助求解技术中，使用神经网络端到端取代传统的数学规划算法（End-to-End Learning）的方法往往存在瓶颈，因为纯数据驱动方法难以保证解的可行性和最优性，且在处理硬约束时缺乏鲁棒性。因此当前AI辅助求解技术趋势是结合深度学习和数学符号推理技术的神经符号混合方案，即利用AI强大的理解归纳能力，从历史求解数据中学习问题结构和决策模式；利用求解器严谨的演绎框架（如B&B树搜索），保证逻辑正确和约束满足。

目前常见的AI辅助求解技术主要可分为求解器超参优化技术、求解器策略嵌入技术和求解器算法发现技术三大方向[83]，这三个方向也对应了AI技术介入求解器程度的逐层递进，整体技术地图如下：

图2.41 AI辅助求解技术地图

1) 求解器超参优化技术：不改变求解器本身的算法逻辑，利用机器学习和贝叶斯优化等技术，针对特定问题实例或实例集自动寻找最优的求解器参数配置，实现针对特定场景的定制化加速。此技术中的超参优化也可称为超参推荐，本文统一称为超参优化。

2) 求解器策略嵌入技术：通过数据驱动的方式，利用图神经网络和强化学习等技术替换或增强求解器内部分支（Branching）、割平面（Cut Selection）等关键模块的决策函数，提升求解器性能。

3) 求解器算法发现技术：利用大语言模型的代码生成与推理能力，自动编写、测试、分析并迭代出全新的求解器算法代码，有望突破现有算法框架瓶颈，实现从“调优”到“创造”的质变。

2.3.1 求解器超参优化技术

2.3.1.1 求解器超参优化简介

现代求解器是高度参数化的软件系统。如开源求解器SCIP包含超过2000个参数，商业求解器如天筹求解器和Gurobi也有数百个可设置的参数项。这些参数控制着求解器全生命周期的核心行为，包括但不限于：

1) 终止与精度控制参数，决定了求解器的终止条件，如：

- 时间限制（TimeLimit）：求解时间到达该阈值后强制停止。
- Gap精度（MIPGap）：当前找到的最佳解与理论最优下界的差距。

- 数值精度（FeasibilityTol）：判断约束是否被满足的容差（避免浮点数误差）。

2) 求解策略参数，决定了求解器使用的求解技巧与策略，如：

- 预处理（Presolve）策略：控制预处理的激进程度、探测（Probing）的深度、特定简化规则的开关。
- 分支策略（Branching）策略：选择变量评分函数的权重（如Strong Branching与Pseudocost的混合比例）。
- 割平面（Separation）策略：决定生成Gomory Cut、MIR Cut、Zero-half Cut等各类割平面的频率、最大轮数及激进程度。
- 原始启发式（Primal Heuristics）策略：控制各类启发式算子（如RINS、Diving、Feasibility Pump）的调用频率和细节配置。
- 线性规划方法（Linear Solver）策略：选择底层LP使用原单纯形法、对偶单纯形法还是内点法求解，以及Scaling策略等。

3) 整体性能参数，决定了求解器整体的策略导向和性能，如：

- 任务重心（MIPFocus）：选择“寻找可行解”和“证明最优性”间的平衡。
- 并行策略（Threads）：调用多少个CPU核心进行并行搜索。
- 随机性（Seed）：设置随机数种子，控制算法的稳定性。

求解器超参优化技术的核心目标是解决指定场景的算法配置优化问题，即：给定一个目标问题实例分布$\mathcal{D}$和求解器$\mathcal{A}$，寻找一组参数配置$\theta^* \in \theta$，使得期望的性能指标y（例如求解时间、Primal-Dual Gap或Primal Integral）最优化。其挑战包括：

- 高度非线性：求解器参数与性能之间是黑盒关系，高度非线性且不可导，无法使用传统梯度下降方法。
- 高维与条件依赖：求解器参数空间巨大且求解器性能对特定参数高度敏感，同时存在大量相互依赖的条件参数。
- 高评估成本：复杂问题的求解时间成本高，可能耗时数十分钟甚至数小时，因此评估一组配置的性能可能成本很高，在缺乏历史求解数据的冷启动场景尤其困难。

2.3.1.2 求解器超参优化软件架构

为了帮助用户和研究人员深入探索求解器的超参数空间，通常可以构建一个

集成多种超参调优技术的超参并行推荐框架，以提高搜索效率，同时也可以通过工具化提升调优操作的便捷性。超参并行推荐框架可以根据搜索空间生成配置（即一组参数），并由求解器进一步进行求解，观测到的结果被用于改进下一步新参数的生成。整个过程不断循环，直到达到最大尝试次数或满足其它停止条件。求解器超参优化软件架构和框架如下所示：

图2.42 求解器超参优化软件架构

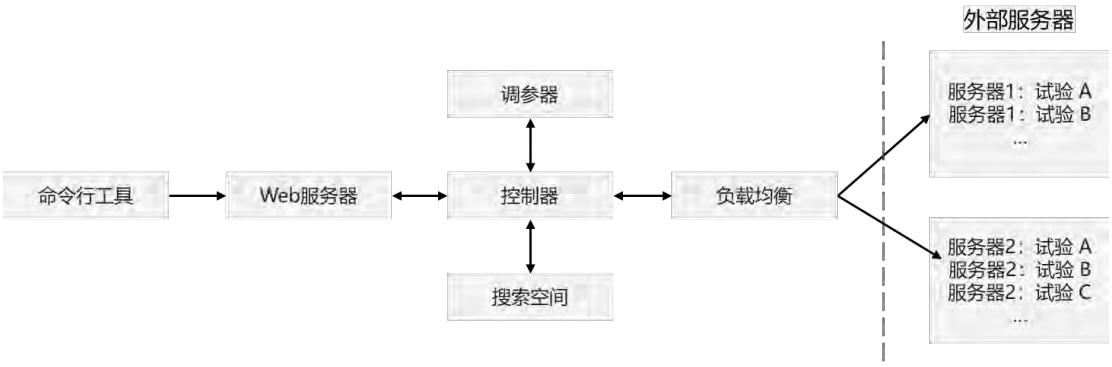

图2.43 求解器超参优化框架示意图

其中，命令行工具用于启动新实验、监控实验执行状态或停止活动实验，并且对接Web服务器；Web服务器可以构建丰富的UI功能，可视化展示实验执行过程中的细节、统计数据以及当前最优解等信息，同时支持从远程服务器收集运行状态；控制器负责协调各模块工作，根据定义的搜索空间，通过调参器生成的策略进行实例化，将抽象的任务转化为具体的超参配置尝试；调参器内置多种超参数优化算法（如贝叶斯优化、随机搜索、TPE等），负责计算并推荐下一组最具潜力的参数组合；搜索空间定义了待优化参数的取值范围与类型；负载均衡模块是任务分发枢纽，需要根据集群的实时负载情况，将控制器生成的测试任务高效地分发到各个可用硬件节点上，确保计算资源的最优利用。整体既支持在单台服务器内并行启动多个试验，也支持跨多台外部服务器进行大规模分布式并行计算。

2.3.1.3 求解器超参优化整体流程

如图2.44所示，超参优化的整体流程可以概括为：配置文件解析、代理模型初始化、新参数推荐、参数评估、代理模型更新、返回结果。整体的核心逻辑是通过迭代的方法，不断推荐新参数再进行评估，平衡“探索”和“利用”，一方面尽量探索未探索过的参数空间，另一方面提升新参数推荐的准确率，在“高价值”参数空间中充分地搜索。不同参数优化算法的实现细节可能有所不同，但是整体流程都与图中展示的基本一致。

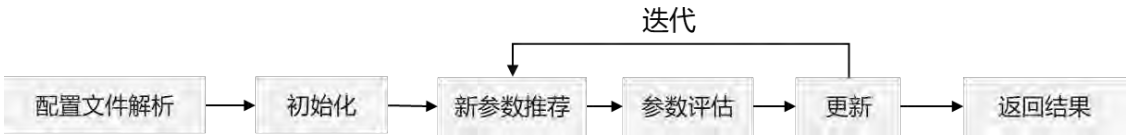

图2.44 超参优化算法流程

2.3.1.4 求解器超参优化功能特性

参数优化技术作为数学规划求解器性能调优的核心手段，应该具有对复杂参数空间进行智能化探索与全方位性能评估的功能。它应支持精细化的搜索空间建模、高效的自动化调参算法、大规模的并行验证以及直观的实验结果分析，从而实现针对特定问题族的最优配置。

1) 参数搜索空间配置：提供精细化的参数定义与管理能力，支持用户或系统自动定义优化范围。支持对连续型（如收敛容忍度）、整型（如启发式调用频率）及离散/类别型（如选择不同的割平面算法）参数进行统一建模。

2) 超参数搜索框架：利用贝叶斯优化、种群算法或者基于大模型的参数调优算法，在参数空间中平衡“探索”与“利用”，以最少的采样次数逼近全局最优参数组合。

3) 分布式并行评估：支持在计算集群中并行验证多组参数，自动解析求解日志，提取关键指标并汇总。

4) 结果可视化分析：支持将高维参数组合与求解性能指标（如Time、Gap、Nodes）通过平行坐标轴进行映射。用户可以直观观察不同参数取值区间与最终求解效率的关联性，发现跨维度的潜在模式。

2.3.1.5 求解器超参优化关键技术

（一）传统贝叶斯优化技术

与黑箱优化求解器技术中的贝叶斯优化类似，在求解器超参优化中贝叶斯优化的执行遵循典型的“采样-评估-更新”循环，大致包含以下步骤：

1) 定义搜索空间：确定待优化参数名称、类型（连续/离散/分类）及其取值范围。

2) 构建代理模型：利用已知采样点构建目标函数（如求解时间、PD Integral）的概率估计模型，捕获预测值及不确定性。

3) 采集函数优化：通过期望改善（Expected Improvement，EI）、改善概率（Probability of Improvement，PI）和上置信界（Upper Confidence Bound，UCB）等采集函数权衡探索（搜索未知区域）与利用（深挖高分区域），产生下一组超参配置。

4) 评估与反馈：调用求解器执行该组超参配置，收集性能数据并反馈给代理模型，完成一轮闭环更新并继续进行下一轮采样，该过程可分布式并行执行。

基于传统贝叶斯优化的一系列改进方法均可用于求解器超参优化，例如通过输入变换来解决数据非平稳性，结合多目标采集函数平衡探索与利用的HEBO算法[84]，曾获得NIPS2020黑箱优化挑战赛冠军；基于Transformer的元优化框架Transformer BO可迁移源任务知识到目标任务，大幅提升调优效率。

（二）大模型驱动的超参优化技术

贝叶斯优化等传统的求解器参数优化技术面临仅关注参数与结果的数值映射，忽视了数学模型背后的语义逻辑，也不能很好地理解并应用已有的求解器参数说明和数学规划求解技术信息，尤其在高求解复杂度、冷启动的使用场景中存在显著瓶颈。大模型技术的引入给这一难题提供了新的技术范式，大模型驱动的超参优化从参数优化逻辑上有了显著革新，通过大语言模型的理解和推理能力不仅能够理解“电力调度”与“物流路径”这样的语义信息在约束结构上的本质差异，也可以通过阅读求解器手册实现冷启动场景的高质量配置。大模型驱动的超参优化框架整体架构可用下图表示：

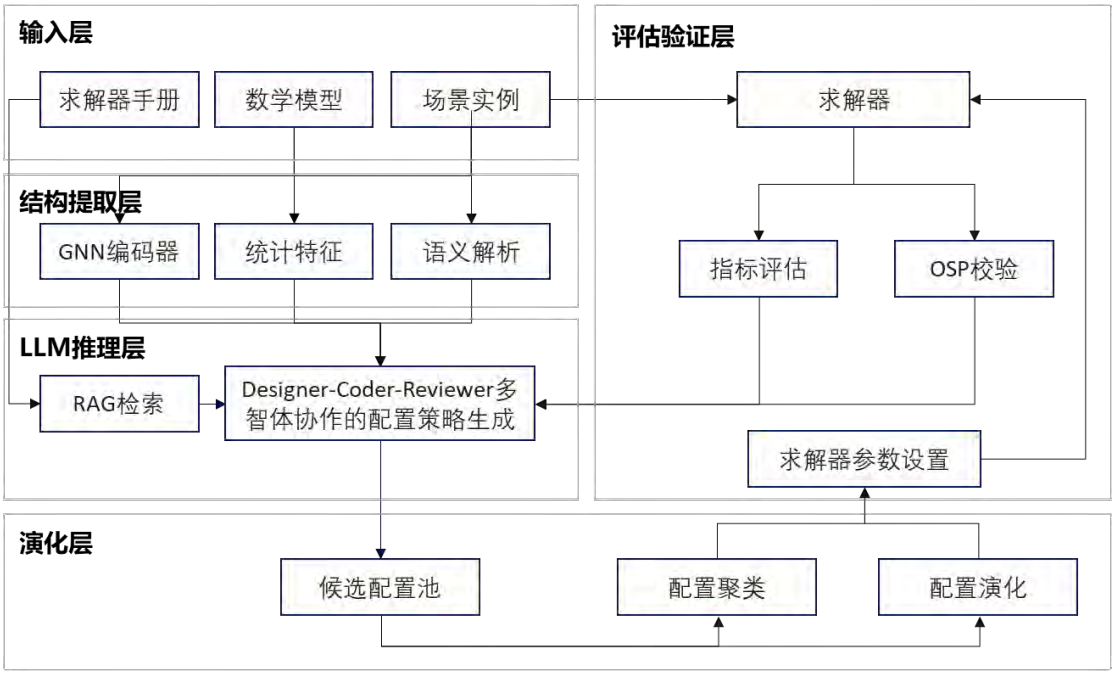

图2.45 大模型驱动的超参优化框架图

其中的核心组件有：

1) 输入层：支持自然语言描述、数学公式以及.mps 文件等不同形式的信息输入。

2) 结构提取层：集成 GNN 编码器将问题的拓扑结构（对称性、稀疏性、退化性）转化为结构嵌入，弥补 LLM 在结构化数据处理上的短板；对于缺乏自然语言描述信息的场景，可以尝试从模型文件中提取约束类型分布，辅助大模型逆向推导问题背景并生成参数。

3) LLM 推理层：参考检索到的相关求解器手册信息，利用提示词工程和多智能体协作框架等技术和所有反馈信息，生成一组推荐的候选配置。

4) 演化层：将 LLM 生成的初始配置放入种群，通过聚类选出最有代表性的差异化配置，并通过 LLM 引导的交叉变异等算子尝试配置领域探索，最终生成本轮迭代需要进行测试的配置集。

5) 评估验证层：将最终选定的测试配置转为求解器参数设置，调用求解器执行实例求解，评估收集求解时间、PD Gap 以及核心过程求解信息等参考指标，并进行最优解保留（OSP）校验，最终将结果反馈给 LLM 推理层辅助生成更优的配置推荐策略。

在涵盖了经典的TSP、Set Covering问题、以及复杂的工业调度问题的混合整数规划问题集上测试求解器割平面超参数配置方法的对比实验[85]表明，通过对传统算法搜索n次（Search(n)）和基于大模型推荐的n个配置（LLM(n)）中择优的结果对比可以看出，LLM(5)策略仅需测试5组配置，就达到了传统搜索方法测试500组配置的同等性能，计算量减小了100倍，整体性能分布也明显优于其他方案。同时大模型推荐的割平面参数配置相比默认配置实现了57.4%的运行时间缩减，即使是大模型直接推荐的参数（LLM(0)）也有近20%的提升，整体体现了极高的数据效率和超参优化能力，可以很好地缓解传统超参优化方法中冷启动场景表现不佳的难题。

2.3.2 求解器策略嵌入技术

2.3.2.1 求解器策略嵌入简介

求解器超参优化技术是在求解器外部进行参数配置，而求解器策略嵌入技术则是深入求解器内部，对其中的关键决策模块进行替换或增强。例如当前MIP求解器的分支定界（Branch and Bound，B&B）算法是一个构建巨大搜索树的过程，在此过程中求解器必须不断做出复杂的启发式决策，包括但不限于：

1) 节点选择（Node Selection）：选择搜索树中的哪个节点进行分支的策略，即树搜索策略，需要权衡发现可行解的速度或提升下界的速度。

2) 分支变量选择（Variable Selection）：在当前节点选择哪个变量进行分裂的策略，该策略会直接决定搜索树的规模。

3) 割平面选择（Cut Selection）：在每一轮生成的数百个候选割平面中，选择哪些加入线性松弛的策略，加多了会由于矩阵变大导致单纯形法变慢，加少了会导致边界收紧不够而缺乏加速作用。

4) 原始启发式（Primal Heuristics）：进行部分解的预测来大幅缩小问题的可行域，从而加速获取可行解来加速整体求解的策略。

传统的求解器依赖人工设计的评分规则来做这些决定，例如强分支（Strong Branching）或伪成本（Pseudo Cost）。求解器策略嵌入技术旨在利用深度学习、强化学习等技术，从数据中学到一个更优的评分函数，替代人工设计的评分规则，在决策质量和计算代价之间取得更优的平衡。由于求解过程具有复杂丰富的特征数

据，各种机器学习技术可以广泛应用于优化节点选择策略中，包括但不限于：

1) 模仿学习：该方法先构建一个理想预言机（例如先用求解器求解全局最优解），再使用监督学习技术（如分类或排序模型）训练一个策略函数，使其对于给定的节点特征，输出的优先级排名尽可能接近预言机的选择。

2) 强化学习：通过求解过程中的反馈（如求解时间或间隙变化）来训练神经网络，使其自主学习如何选择策略，从而在求解过程中提供更准确的评估判断。

3) 图神经网络：利用MIP问题的二分图表示（节点代表变量和约束，边代表系数），使用GNN来比较不同节点的潜力或估计变量值，进一步指导相关策略选择。

4) 符号发现：尝试寻找一套可解释的数学表达式来替代神经网络，同时可以提高最终得到的策略在CPU上的计算效率。

2.3.2.2 求解器策略嵌入软件架构

整个策略嵌入的软件架构和训推过程分别如图2.46和图2.47所示，AI算法通过回调函数跟求解器进行交互，收集求解过程中的<State,Action,Reward>来训练强化学习模型，最后训练好的模型会保存成一个.opm（**O**ptVerse **P**retrained **M**odel）格式的文件。在求解新问题时候，可以通过命令行参数把该模型传给求解器，求解器内会自动解析这个文件，自动把相关的AI模型嵌入到求解中的指定模块，这样在求解时候就会用AI模型推理得到的结果来做求解决策。

图2.46 求解器AI策略嵌入软件架构

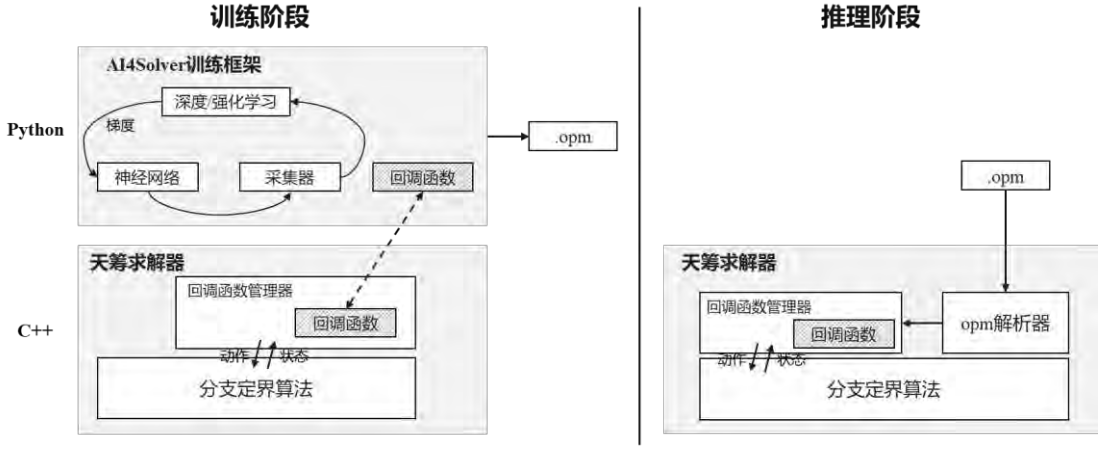

图2.47 求解器AI策略嵌入训推流程图

2.3.2.3 求解器策略嵌入整体流程

图2.48展示了求解器策略嵌入的整体流程，根据使用的学习算法不同，流程也略有区别。模仿学习需要提前收集好样本的特征和标签组成$\langle x, y\rangle$数据对，然后再进行模型训练，这里的模型可以神经网络或者是传统机器学习的树模型。强化学习的训练需要在线和环境（求解器）进行交互获取奖励值来更新模型权重，因此需要利用求解器回调函数把策略模型输出的决策值传给求解器，影响他求解过程中的决策，并且通过接口获取求解过程中的短期奖励和长期奖励值。最终两种方法都会生成一个统一格式的模型文件，用于在测试阶段被求解器调用。

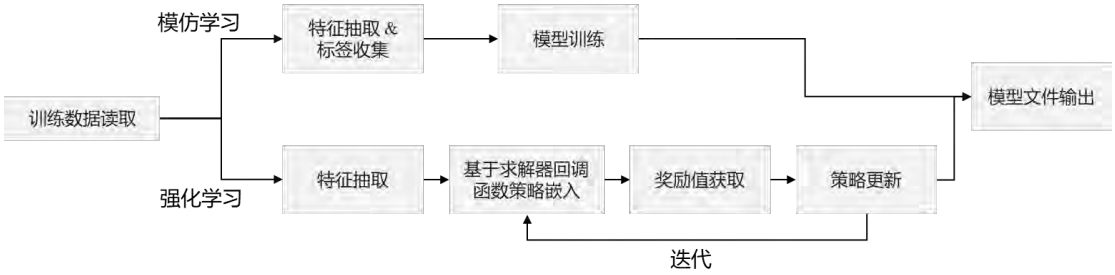

图2.48 求解器策略嵌入整体流程

2.3.2.4 求解器策略嵌入功能特性

求解器策略嵌入作为连接数据驱动决策与传统优化算法的核心桥梁，应该具有对求解器内核进行深度干预与智能赋能的功能。它通过构建标准化的特征转化机制、实时的状态反馈链路以及统一的模型交互协议，实现对求解器内部关键决策模块的智能化替代与性能重塑。

1) **特征抽取模块（Feature Extraction Module）**：提供高效、标准化的数据表征能力，将原始数学模型与求解状态转化为 AI 模型可识别的特征向量或图结构。

- 结构化特征提取：支持将约束矩阵转化为变量-约束二部图（Bipartite Graph），捕获问题的拓扑结构特征。
- 多维属性编码：自动提取变量的类型、上下界、目标函数系数，以及约束的右端项、灵敏度信息等静态属性。
- 动态特征融合：实时捕捉求解过程中的统计特征（如节点深度、伪成本、LP 残差），并将其与结构特征进行融合。

2) **中间状态实时感知**：具备深度的内核探测能力，能够实时获取分支定界过程中的**动态运行快照**。包括但不限于：活跃节点分布、全局上下界的收敛轨迹、基矩阵的状态、以及历史分支路径的成功率。

3) **基于回调函数的决策干预（Callback-driven Decision）**：支持通过高性能回调接口将 AI 策略嵌入求解生命周期，接管核心搜索决策：

- **分支回调**：调用 AI 模型预测最具潜力的分支变量，取代传统的强分支策略，以减少搜索树规模。
- **启发式回调**：根据当前状态动态触发神经启发式算法，直接为求解器提供高质量的整数初始解。
- **剪枝与割平面回调**：由模型评估当前节点的获益潜力，决策如何筛选高效割平面。

4) **统一模型文件格式（Unified Model File Format）**：为了确保不同框架（如 PyTorch, TensorFlow）训练的 AI 策略能稳定运行在求解器内核中，应支持标准化的模型交换格式。

- 工业级标准支持：全面兼容 ONNX 格式，确保模型在跨平台、跨语言（如 C++ 加载 Python 训练的模型）环境下的高性能推理。
- 元数据管理：模型文件应包含输入输出张量的维度定义、预处理参数、模型版本号及适用的问题类型标签，确保模型加载的准确性。

2.3.2.5 求解器策略嵌入关键技术

（一）节点选择

在分支定界算法B&B中，求解器会维护一个存有待处理子问题的活动节点队列。节点选择是指从该队列中重复挑选出一个节点进行探索的策略。其关键问题在于如何确定待处理节点的优先级，以引导搜索树向最有希望的区域探索，从而尽快找到优秀的可行解，因此被认为是B&B中最关键的决策之一。

通过AI嵌入求解器节点选择策略可以融合多种启发式特征，并从中学习到适应特定问题结构的复合规律，从而有效引导搜索树向最有希望的区域探索，显著减少搜索节点数，实现整体求解加速。例如文献[86]通过对SCIP内置节点选择策略的模仿学习，可以获得难例探索效率更高、泛化性更强的节点选择策略。该策略在求解时间上与原策略相比无统计差异，但在有限时间限制下解质量显著更优，相同时间内最优间隙/整数间隙远低于SCIP及其他基线，在难例中优势更突出。

（二）分支变量选择

分支变量选择旨在选择合适的分支变量以缩减搜索树规模，同样也可以用模仿学习方法实现，例如通过混合GNN架构和优化的训练协议，FiLM方法[87]对效果好但计算成本过高的强分支策略在搜索树中的决策行为进行了模仿学习，最终能够实现在不完全信息下也能优先扩展包含最优解的节点，或提前剪掉不具潜力的分支。数值实验结果表明，在多个不同类型的混合整数线性规划问题库的实验中，这种基于模仿学习的搜索策略都保持了较高的预测准确率，能够有效地指导B&B的树探索。

同时，也可以进一步利用符号发现方法学习轻量级且可解释的数学表达式作为分支策略，并且可以避免复杂GNN在线推理的成本。例如基于符号发现的Symb4CO框架[88]通过模仿强分支策略训练，仅使用10个训练实例就能训练出SOTA水平的符号策略，实验显示其准确率远超可解释的Trees模型，与复杂MLP模型相当。另一个借助组合优化问题二部图信息的GS4CO框架[89]同样通过模仿强分支策略学习到符号策略，在纯CPU环境下超越所有基线方法，并且推理速度比传统的图神经网络快约20倍、模型体积仅为约1/500，在工业级部署中有显著的优势。

（三）割平面选择

在实际的混合整数规划问题中，线性松弛给出的界限往往很松。如果不加割平面限制，B&B可能会在接近最优解的位置大量迭代难以收敛。割平面选择中的核心要处理的“选多少割平面”、“选哪些割平面”以及“以什么顺序加入割平面”三大问题，可以通过强化学习训练分层序列模型（HEM）同时解决[90]。HEM利用双层级联结构的指针网络捕捉割平面之间的相互作用，高层决定选多少割平面、底层决定具体选择的割平面及其顺序，通过注意力机制同时捕捉割平面之间的顺序依赖和交互关系以避免冗余。以求解效率为奖励，通过梯度上升最大化期望奖励，分别更新高层和底层的参数，实现端到端学习。若进一步将单轮强化学习扩展为多轮MDP，重新定义动作为“选割比例+有序割子集”的组合，将底层基于LSTM的序列到序列模型替代为基于多头注意力编码器的集合到序列模型以去除输入序列的冗余顺序信息，并且优化训练策略，可以得到增强的分层序列模型（HEM++），弥补HEM在多轮割选择、输入顺序冗余、样本效率上的不足。

实验结果表明，HEM在9类基准上降低求解时间30%~80%，显著优于手工启发式和现有学习方法。并且在超过训练数据规模2~9倍的实例上，HEM仍能保持稳定优势，证明其优秀的鲁棒性、泛化性及对大规模问题的适配性。同时在华为生产计划和订单匹配两大真实场景数据测试中，HEM能够提升求解效率生产计划问题求解时间13.28%，订单匹配问题求解时间82.34%，效果显著[90]。

此外，如何使用各种分离器即生成割平面的算法的问题也可以通过强化学习技术来解决。例如动态分离器配置方法DynSep[91]将每轮的分离器配置任务建模为序列决策问题并通过强化学习技术进行策略优化，其核心在于：将状态表示增量三元图以捕捉每轮切割后新添加的割平面约束、收紧的变量边界和更新的分离器统计，重新设置了平衡求解效率与松弛域收紧效果的增量奖励函数，并使用图卷积网络Token化并使用PPO算法训练带块位置编码的Decoder-only Transformer。实验表明该方法能显著提升问题端到端求解效率，并同样能在远大于训练规模的实例上仍保持优势[91]。

（四）原始启发式

通过设计合适的启发式快速找到一个可行解可能通过高效剪枝极大地促进原问题的求解，例如Diving方法会执行深度优先的搜索，尽快找到一个整数解。而通

过AI技术可以高效辅助类似的启发式算法设计。例如Neural Diving技术[92]是利用图卷积神经网络提取数学模型二分图的结构特征，进而通过预测二进制变量的取值概率（即解的分布），识别出预测置信度较高的变量子集并将其直接固定为预测值，可以构造出一个规模大幅缩减的子问题，再调用求解器求解子问题即可能获得一个最终的可行解。实验表明Neural Diving技术能够极大地缩减搜索空间，相比传统求解器在多个合成与真实数据集上实现了3到10倍的求解加速效果。

在此基础上，Predict-and-Search框架[93]用GNN预测每个变量的边缘概率分布替代Neural Diving的离散赋值，通过在预测解的信任区域中进行邻域搜索替代直接固定变量，可能提供更全面的解空间信息并在可行性与最优性间取得理论上更优的平衡。Apollo-MILP框架[94]则进一步构建了多轮迭代的“预测→修正→降维”闭环：在每一轮的预测步骤中定位当前实例的未固定变量，用GNN预测其边际概率分布；在修正步骤中通过信任区域搜索得到“参考解”，进一步提出不确定性误差上界（UEBO）方法筛选可靠变量；继而在降维步骤中固定预测值与参考解值相等的低UEBO值变量，从而生成下一轮的简化实例。实验证明Apollo-MILP在多个数据集上都能有效降低gap值，并在真实工业复杂场景（如电力、调度）实例中仍保持稳定优势。

（五）多算法协同技术

现有基于机器学习的方法往往仅优化单个求解器模块（如分支或割平面选择），忽略了模块间的相互依赖（例如割平面选择影响LP松弛强度，进而改变分支决策），可能导致整体优化效果受限。因此，采用多算法协同优化框架可能进一步提升求解效率和质量。例如Collab-Solver[95]是基于多智能体的协同策略学习框架，重点解决割平面选择与分支变量选择模块的协同优化，采用了“两阶段学习+Stackelberg博弈建模”实现两个模块的协同，其核心流程包括：

1) 问题建模：Stackelberg博弈

将割平面选择（作为领导者）与分支变量选择（作为追随者）建模为有限步部分可观测马尔可夫博弈问题$G = (S, A_c, A_b, P, O_c, O_b, r)$，其中的核心要素是：

- **状态与动作空间**：S为MILP求解状态，A_c是割平面选择动作、A_b是分支变量选择动作，A为智能体动作空间。

- **观测空间**：O_c是割平面相关特征，O_b是以二分图形式表示的 MILP 实例特征。
- **优化目标**：最小化平均求解时间（用于简单实例）或原对偶间隙（PD gap，用于困难实例），目标函数为：

$$\min_{\pi_c \in \Pi_c} \mathcal{T}_{\mathrm{D}}(\pi_c, \pi_b^*)$$

$$s.t. \pi_b^* \in \arg\min_{\pi_b \in \Pi_b} \mathcal{T}_{\mathrm{D}}(\pi_c, \pi_b)$$

其中$\mathcal{T}_{\mathrm{D}}(\pi_c, \pi_b)$为训练集$D$上的期望求解代价。

2) 两阶段学习范式

阶段1：数据通信预训练，通过数据通信实现两模块初步协同，避免孤立训练，即：

- **割平面选择策略（π_c）预训练**

输入：割平面特征O_c（13维向量，含系数统计、有效性等）+MILP特征O_b（二分图，含变量/约束属性）。

网络结构：LSTM编码割平面序列特征 +GCNN编码二分图MILP特征 + 交叉注意力融合 + 指针网络处理变长度动作空间。

优化策略：强化学习策略梯度，奖励为负求解时间（简单实例）或负PD gap（难例）。

- **分支变量选择策略（π_b）预训练**

数据通信：用预训练得到的π_c替换求解器默认割平面选择策略，生成受到割平面选择影响的分支场景。

专家数据：通过计算成本高但性能好的强分支策略（生成最小搜索树）作为专家方案构建高效率的分支决策数据集D_e供模仿学习。

优化策略：最小化行为克隆的损失$L(\psi) = -\frac{1}{N}\sum \log\ \pi_b(a_b^i \mid o_b^i)$，实现神经网络推理级的实时分支决策速度。

阶段2：并发联合微调，解决协同训练中的非平稳性问题，采用双时标的更新规则：

- 慢时标（领导者π_c）：每ω_c=4 个实例更新 1 次，由于割平面选择对全局求解影响更大，需更稳定的更新。

- 快时标（追随者π_b）：每ω_b=1 个实例更新 1 次，快速适配π_c的变化。
- 优化策略：均采用策略梯度更新，轨迹由π_c和π_b交替共同生成，奖励设计为最大化双方协同后整体的求解效率。

3) 实验结果

Collab-Solver在多个基准数据集上进行了验证[95]，其结果无论在求解时间还是PD Integral指标上均优于其他方法。不仅相比基准求解器性能提升了60%以上，同时也稳定超过了仅使用单个AI嵌入模块的算法HEM和GCNN-B。由此可以证明，这类多算法协同的优化技术的有效性。

2.3.3 求解器算法发现技术

2.3.3.1 求解器算法发现简介

基于大语言模型的自动算法设计（Large Language Model for Algorithm Design, LLM4AD）是当前优化技术与人工智能交叉领域的前沿方向。其核心思想是将大模型作为“进化算子”或“策略生成器”，利用其强大的代码生成能力和逻辑推理能力，自动编写、探索和优化算法中的关键组件。传统的自动算法设计技术（如SMAC、IRACE）主要是在算法的超参数空间进行搜索，基于大模型的算法设计可以在代码逻辑空间进行搜索。LLM4AD的算法的核心特征是：

1) 代码级搜索空间：LLM 直接生成可运行的算法代码（C++/Python/Java），比单纯调参具有更高的自由度。

2) LLM 作为进化算子：LLM 充当了进化计算框架中的变异和交叉算子的角色，根据历史表现最好的代码片段，通过语义理解生成改进版本。

3) 闭环反馈：系统自动将算法在 Benchmark 上的运行结果（如耗时、Gap 值、搜索节点数）得到评价指标的反馈，形成“生成-测试-学习-改进”的闭环。

求解器自动算法发现技术也是通过大语言模型与进化算法的有机结合，实现求解策略的自动迭代与进化。根据代码的接入深度与作用机制，该技术体系分为回调接口驱动模式与源码级深度演化模式。

（一）回调接口驱动模式

以天筹自动化算法设计平台OptvEvolve为例，回调接口驱动模式通过天筹AI求

解器提供的标准化API接口，实现外部算法演化逻辑与求解内核的轻量化耦合，如图2.49所示：

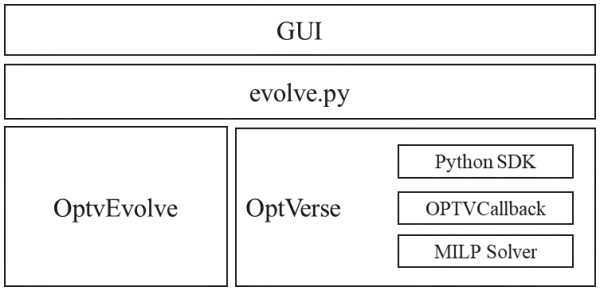

图2.49 基于OptvEvolve的求解器算法发现系统

架构协同机制：架构底层由自动化算法设计平台与天筹数学规划求解器构成。天筹求解器开放了涵盖分支定界、割平面生成及原始启发式等关键环节的回调函数（Callbacks）。通过这些接口，系统能够将LLM生成的定制化策略动态注入求解流程，实现针对特定问题特征的精准加速。

任务配置与闭环演化：演化流程由驱动脚本evolve.py和配置文件config.yaml统筹定义。其中不仅包括了评估基准和演化算子，还深度集成了Prompt工程模板、LLM API信息及种群遗传策略等，该框架据此进行“生成-编译-测试-反馈”的自动化闭环实现算法的自主迭代。即在每一轮演化中，系统依据评估基准对当前种群进行多维度打分，利用LLM进行语义分析并提取特征反馈，从而动态调整Prompt诱导策略，在复杂的求解空间中持续探索更优的算子组合。

可视化监控与分析：演化任务启动后，用户可通过前端监控平台实时追踪种群多样性、适应度（Fitness）收敛曲线及最优算法片段的演进过程。

（二）源码级深度演化模式

源码级深度演化模式实现了更高维度的自动化设计，即直接介入求解器的底层逻辑，实现对核心算法架构和代码的深度重构与自主优化，其特点在于：

- 演化对象深度：与回调模式不同，该模式的演化对象深入至求解器内核的核心代码片段。通过对底层算子逻辑、搜索树管理或数值计算模块的直接重构，甚至能够探索非直觉的优化路径。
- 技术挑战与优势：该模式对代码的正确性验证和编译稳定性提出了极高要求，但其优势在于不再受限于预定义的接口，而是具备了从底层逻辑层面重塑

求解器行为的能力，因而能够实现更彻底的算法发现，适用于对求解性能有极致追求的垂直行业场景。

下表总结了两种方式在不同维度上的区别：

表2.3 回调接口驱动模式与源码级深度演化对比

维度	回调接口驱动模式	源码级深度演化模式
集成深度	外部逻辑注入，低耦合	底层内核重构，高耦合
演化目标	启发式策略、参数调优	核心算子逻辑、底层框架
开发难度	较低，受限于预设接口	较高，需保障内核稳定性
性能上限	针对特定环节的局部优化	具备实现突破性算法发现的潜力
典型场景	针对特定行业算例快速适配	通用求解器性能攻坚与架构创新

下面通过基于大模型的自动算法设计技术在求解器两个模块上的应用案例介绍该技术的细节。

2.3.3.2 求解器算法发现软件架构

求解器算法发现软件架构如图2.50所示，整体算法发现框架分成了5层。

图2.50 基于大模型的算法发现软件架构

1) 基础设施层：负责支撑上层算法发现的配套模块，包括日志管理，数据集管理，并行调度器管理等；

2) 大模型层：用于跟大模型进行交互，比如直接访问大模型的客户端，或者是

调用代码智能体（如OpenCode）的接口；

3) 核心层：用于实现算法自演化的核心模块层，包括了算法思想生成、算法编码、算法评估的全流程，利用搜索算法进行迭代优化；

4) 配置层：整个系统所有配置参数的管理；

5) 交互层：用于跟用户进行交互，包括SDK，CLI和可视化的实验管理。

2.3.3.3 求解器算法发现整体流程

求解器算法发现整体流程如图2.51所示，整体分成了基线算法分析，种群初始化，新算法采样，算法构建，算法评测，种群更新，结果输出这几个模块。整体的流程目的是利用大模型的代码生成能力自动生成求解器中部分的算法代码，并且通过上层的搜索框架给大模型提供优化的方向，最终通过迭代搜索到一个在数据集上表现较好的新算法。目前大多数基于大模型的算法发现技术都是基于种群的演化学习算法，流程与图中所示的流程基本一致。

图2.51 求解器算法发现整体流程

2.3.3.4 求解器算法发现功能特性

求解器算法发现作为推动求解引擎自主演进的前沿技术，应该具有从算法评测反馈中提炼经验并将其自动化转化为高性能算子的功能。它通过整合大语言模型的生成能力与演化算法的迭代机制，构建起一套从逻辑构思到代码落地、再到性能闭环进化的全生命周期自动研发体系。

1) **求解经验自动提取**：利用 LLM 自动扫描并解析海量求解日志、科研论文及开源代码库。系统能够识别特定算子在不同问题上的失败模式，并自动总结出改进策略的自然语言描述，将其转化为代码生成的 Prompt 指导信息。

2) **自动化算法代码合成**：支持基于 LLM 自动生成符合求解器架构约束的底层算法代码（如 C++ 预处理规则、LNS 邻域算子等）。生成的代码需经过严格的语法检查与静态分析，确保内存安全与数值计算的稳定性。

3) **分布式算法性能评估**：

- 集群化测试：构建基于容器化的分布式评测矩阵，将 LLM 生成的多个算法版本分发至高性能计算节点，在多类型的测试集（如 MIPLIB、QPLIB）上执行并行基准测试。
- 统计学置信度评价：自动汇总分布式测试数据，通过置信区间、加速比分布等统计指标，排除随机干扰，客观评价算法改进的有效性。

4) **演化学习与迭代优化**：引入模拟自然进化的迭代机制，以 LLM 作为“变异算子”对现有代码进行逻辑重组。

- 代码种群演进：优选性能表现优异的算法片段进入“精英池”，通过交叉、变异产生下一代候选算法。
- 闭环自纠错：将分布式评估中的错误日志及性能指标实时反馈给 LLM，驱动其进行针对性的逻辑修复与性能优化，实现算法逻辑的持续自我增强。

2.3.3.5 求解器算法发现关键技术

（一）启发式算法发现技术

在处理混合整数规划问题时，原始启发式是求解器在分支定界框架下寻找可行解的核心机制。其主要目标是在搜索树的早期或特定节点，通过非完备性搜索快速捕捉高质量的整数可行解。这不仅能为全局优化提供有效的上界（Upper Bound），更能显著加速剪枝过程，从而缩短整体求解周期。

例如下潜（Diving）启发式是一类极其重要且高效的构造启发式算法。其核心思想是模拟分支过程中的“深度优先搜索”策略：

1) 松弛求解：首先求解当前节点的线性规划（LP）松弛。

2) 变量固定：根据特定的选择规则（如分数值最小、伪成本最高等），选择一个非整数值的变量，并将其强制固定为相邻的整数值。

3) 迭代推进：重新求解由于固定变量而产生的新LP，循环往复，直到找到一个整数可行解或证明当前路径不可行。

Diving启发式在现代商业求解器中占据举足轻重的地位，主要原因为：

1) 寻解能力强：相比于简单的舍入（Rounding）技术，Diving通过不断重新求解LP考虑了变量间的耦合关系，生成的解质量通常更高。

2) Gap收缩效率高：在处理大规模工业规模问题时，尽早发现一个较优的可行

解能够极大地压缩Primal-Dual Gap，防止内存因搜索树过大而溢出。

3) 策略多样性：Diving并非单一算法，而是包含Fractional Diving、Vector Diving、Coefficient Diving等多种变体，为应对不同特征的问题提供了丰富的策略空间。

以对Diving启发式进行自动演化的框架LLM4Solver[96]为例，其中的核心设计点包括：

1) 算法表示与适应度评估

个体表示：每个进化个体包含“下潜启发式评分函数（Python格式）”和“逻辑描述”。评分函数以变量的13个特征（如分数部分、伪成本、目标函数值等，覆盖SCIP人工启发式所用特征）为输入，输出变量评分（float）和rounding方向（bool）；逻辑描述辅助LLM和人类理解算法，指导新算法生成。

适应度计算：将评分函数嵌入求解器中并关闭其他启发式，在根节点执行潜水操作，以“相对原始间隙”的均值作为适应度（值越小，算法性能越好），公式为：

$$f(s) = mean(\gamma_P(\tilde{z}_1^s), \gamma_P(\tilde{z}_2^s), \dots, \gamma_P(\tilde{z}_N^s))$$

其中$\gamma_P(\tilde{z})$为当前可行解与全局最优解的目标值差距比例。

2) 基于LLM的新个体生成

通过提示工程（Prompt Engineering）引导LLM在初始化、交叉、变异步骤生成算法，提示分为两类：

- 背景提示：提供 MILP 定义、下潜启发式原理、通用下潜伪代码等领域知识，帮助 LLM 理解任务背景。
- 任务特定提示：明确输入特征描述、输入输出格式（如函数名、13 个输入参数、输出“评分+rounding 方向”）及任务指令（如初始化需从零生成算法，交叉需融合父算法优势）。
- 具体生成步骤：

初始化：用LLM生成N个个体，构成初始种群$P = \{s_1, s_2, \dots, s_N\}$。

交叉：选择父算法，通过LLM融合其优势，生成r个后代个体。

变异：用LLM对交叉生成的后代进行微调，探索其附近的新解。

3) 进化优化策略

父选择：采用适应度比例选择，平衡随机性与最优性，公式为：

$$g(s_n) = \frac{\frac{1}{f(s_k)+\epsilon}}{\sum_{k=1}^{N}\frac{1}{f(s_k)+\epsilon}}$$

其中ϵ是避免分母为0的小数，性能越好的个体被选为父代的概率越高。

存活选择：精英存活选择，从所有父代个体和后代个体中，选择适应度最优的N个个体进入下一代，保障种群最优性。

4) 多目标进化（MOEA）提升泛化能力

目标定义：将算法在不同基准上的性能作为独立目标（如Setcover、Cauctions等4个基准对应4个目标），目标函数为：$f_m(s) = mean(\gamma_P(\tilde{z}_1^s), \gamma_P(\tilde{z}_2^s), \dots, \gamma_P(\tilde{z}_N^s))$

选择策略：父选择采用二元锦标赛选择（随机选2个个体，再选更优者）；存活选择结合非支配排序（按被支配程度分秩，秩越低越优）和拥挤距离排序（鼓励种群多样性）方法，最终从帕累托前沿（Pareto Front）中选择跨基准性能最优的算法。

基于天筹求解器，大模型驱动的自动算法设计LLM4Solver在Diving上的核心实验结果如下：

1) 解质量：显著优于现有方法

在4个标准基准上，LLM4Solver（单目标进化SOEA）的平均相对原始间隙远低于人工设计和L2DIVE。例如，Setcover基准中，LLM4Solver间隙为3.36（0.25），最佳人工设计为6.99（0.38），L2DIVE为3.58。

2) 求解效率：大幅提升求解器性能

- 在 LoadBalance 上，LLM4Solver 使原始-对偶积分较默认求解器降低 38%、较参数调优后的求解器降低 15%，获胜实例数达 99/100。
- 在 NNVerify 上，LLM4Solver 使求解时间较默认求解器减少 31%、较参数调优后的求解器减少 20%，获胜实例数达 376/523（默认求解器为 44）。

3) 搜索效率：快速收敛

LLM4Solver在第1代即可超越最佳人工设计算法，第4代超越L2DIVE，10代内收敛。Setcover上基准10代收敛时间仅3503.5±73.4秒。

4) 跨基准泛化能力：多目标进化优势显著

- 单目标进化（SOEA）在训练基准上性能优异，但泛化到其他基准时性能骤降（如 Indset 训练的算法在 Setcover 上间隙达 70.96）。
- 多目标进化（MOEA）在所有基准上均优于最佳人工设计，且在 MIPLIB2017 异质实例上表现突出：平均原始间隙 844（最佳人工设计为 1180），获胜实例 8/18（最佳人工设计为 4/15）。

（二）割平面算法发现技术

割平面是一类用于强化线性松弛（LP Relaxation）质量的有效不等式（Valid Inequalities）。在求解整数规划问题时，松弛后的线性解往往落在整数可行解形成的凸包（Convex Hull）之外。割平面技术通过在搜索过程中动态挖掘并加入额外的线性约束，在不剔除任何整数可行解的前提下，“切割”掉包含当前分数解的非整数区域。其在求解器架构中重要性体现在：

- 紧缩松弛边界：通过不断迭代添加割平面，使松弛问题的解空间持续逼近整数凸包，从而获得更紧致的对偶下界（Lower Bound）。
- 降低分支压力：高质量的割平面能显著减少分支定界树的节点总数。在许多复杂算例中，仅靠割平面阶段就能将原始 Gap 缩小 50%以上，避免求解器陷入指数级的穷举搜索。
- 数值稳定性提升：优秀的割平面生成算法（如 Gomory 割、混合整数舍入割等）能够优化模型的约束结构，提高数值求解的鲁棒性。

例如，自动化割平面生成框架EvoCut[97]的核心流程如下：

1) 数据预处理（Data Pre-processing）

构建两个数据集，为后续验证和评估提供基础：

- 评估集（De）：默认 10 个实例，运行基线整数规划模型，记录“参考最优性差距”（gapref(i)），作为衡量割平面效果的基准。
- 验证集（Dv）：默认 2 个实例，额外记录两项关键信息：整数规划的最优解（$\hat{x}_i^*,\hat{y}_i^*$），用于验证割平面是否保留最优解；线性松弛（LP）的最优解（x_i^{LP},y_i^{LP}），用于验证割平面是否能“切断”分数解（即有用性）。

2) 种群初始化（Population Initialization）

由LLM驱动的“初始化智能体”生成初始候选割平面种群，流程如下：

- 输入：基线整数规划模型代码（如 Pyomo）、自然语言问题描述、已生成的割平面（避免重复）。
- 生成与验证：LLM 输出割平面的“逻辑说明”和“可执行代码”，需通过以下 3 轮验证：

代码检查：编译代码，排除语法/运行时错误；

最优解保留（OSP）检查：将割平面加入基线模型，固定最优解后验证可行性，确保不排除最优解；

有用性检查：将割平面加入线性松弛模型，固定LP最优解后验证不可行性，确保能切断分数解。

- **终止条件**：生成预设规模（默认 8 个）的有效候选割平面，进入进化阶段。

3) 演化过程（Evolution）

通过多轮迭代（默认20代）优化割平面，在每轮迭代中需要执行的核心操作见下表：

表2.4 EvoCut中的核心演化操作

操作	**逻辑**
精英保留	保留每代中fitness前20%（默认比例re=0.2）的优质割平面，直接进入下一代。
选择	按fitness比例选择父代割平面，fitness越高，被选中概率越大。
交叉（Pc=0.7）	随机选择“交叉智能体”（共4种，如交集交叉、互补交叉），融合两个父代的结构/逻辑，生成子代割平面。
变异（Pm=0.3）	随机选择“变异智能体”（共3种，如提升变异、探索性变异），修改父代的系数/项，生成子代割平面。
验证与评估	子代割平面需重复“初始化阶段”的3轮验证，通过后计算fitness，加入下一代种群。

4) 关键指标：Fitness计算

量化割平面对求解效率的提升的公式如下：

- 单实例相对差距变化：的$d(i) = \frac{gap_{cut}(i) - gap_{ref}(i)}{gap_{ref}(i)}$（负数值表示差距缩小，即割平面有效）；
- 平均差距变化：$c = \frac{1}{|D_e|}\sum_{i \in D_e} d(i)$

- Fitness 映射：$Fit(C) = 10e^{-c}$，差距缩小越多（c 越负），Fitness 越高。

5) 核心实验结果

- **最优性差距减小**

如下表所示，在300秒时间限制下，EvoCut生成的割平面能显著缩小最优性差距，且能泛化到未观测过的实例上：

表2.5 割平面自动生成技术在相同时延下的提升效果[97]

问题	300秒差距 reduction（%）	OSP率（%）	说明
TSP	57.4±26.3	100	效果最优，差距缩小超一半
CWLP	46.2±41.1	100	后期加速明显
JSSP	37.3±22.0	100	早期差距缩小快
MCND	17.1±20.2	100	虽最低，但仍有显著提升

注：OSP率100%表示所有测试实例中，割平面均保留最优解，实际效果等同于“理论最优保留割平面”

- **求解速度提升**

以TSP问题为例，EvoCut生成的割平面方法使达到相同目标差距的时间显著缩短，最高提速4倍。

表2.6 割平面自动生成技术在相同目标下的提升效果[97]

目标差距	平均时间节省（%）	对应提速倍数
10^{-4}（最优）	51±31	~2倍
10^{-1}（10%）	74±48	~4倍

3 领域求解器关键技术

领域求解器（Domain-Specific Solver）是面向特定行业复杂决策难题的专用优化引擎。它将通用数学规划与优化算法能力，同具体业务场景的深层逻辑与领域约束相结合，从而高效求解供应链多环节和领域中的NP难问题。因此，领域求解器完成了从“难以直接应用的通用计算工具”到“能精准破解业务难题的高效智能应用”的关键蜕变，成为连接抽象算法与产业实践的核心赋能组件。

本章重点介绍天筹领域求解器聚焦的供应链与生产运营中的核心优化场景，针对特定行业复杂决策问题而构建的核心优化引擎。天筹领域求解器集成了五大智能决策工具：

1) 批量计划求解器实现产能与订单的精确平衡。

2) 生产调度求解器优化工序排序与资源分配。

3) 运输计划求解器最小化物流成本与时间。

4) 二维切割最大化原材料利用率。

5) 三维装箱求解器最大化装载效率。

各求解器深度融合领域知识，运用数学规划、启发式算法等先进技术，将复杂的业务约束转化为可计算模型，为企业从生产到配送的全链路提供精准、高效的成本与效率最优解。这些模块既可独立应对专项问题，亦可协同工作，形成覆盖“计划-生产-物流”一体化的决策大脑，驱动运营智能化升级。

3.1 批量计划求解技术

3.1.1 批量计划应用背景

现代制造业的生产计划复杂性源于多层级产品结构、多品种混合生产与资源约束的多重限制，需在预定周期内协调人力、设备、原料等资源，平衡库存短缺、生产切换决策（如设备维护与模具更换）、替代方案及环境合规成本（如排放控制）等关键因素。经典批量生产问题（如单产品容量约束模型）通过引入多层级BOM、

可变切换时间、加班成本及替代特性，逐步扩展为多级多资源约束生产批量问题（Multi-Level Capacitated Lot Sizing Problem ， MLCLSP），其决策核心为各时段产品的生产类型与数量，被视为物料需求计划（Material Requirement Planning，MRP）的理论基础。在实际场景中，如大型空调制造厂商的钣金冲压产线，需应对原料供应波动、设备切换损耗、库存周转效率与交付周期的动态平衡挑战，传统经验排程难以满足资源能力（人力、模具、能源等）的严格约束，导致成本高企与交付风险。

在多级生产优化场景中，决策体系围绕生产结构、资源分配与成本控制三大核心维度构建。生产结构决策聚焦原材料、半成品及终端产品的多级生产路径规划与转换率优化，同时涵盖替代生产方案（如生产/交付替代量）的智能选择。资源分配需统筹机器班次类型、分时段加工任务与产量规划、模具动态调配，并基于瓶颈预测弹性调整产能扩容量（机器/模具）。成本控制需综合平衡启动成本（换型、班次切换）、库存成本（持有与过期损耗）、额外资源成本（外购产能/模具）及违约成本（延期或缺货），形成全局成本最优策略。关键约束变量包括影响交付可行性的生产提前期、库存容量限制下的库存/缺货/替代平衡、模具匹配规则（换模效率与兼容性）以及资源冻结期对产量调整窗口的制约。所有决策需在满足多级BOM约束、资源能力上限及业务规则的前提下，实现成本最小化与交付风险控制的协同优化。

针对多级批量计划问题的求解，需首先基于多级BOM结构、资源属性与业务规则构建数学模型，明确生产量、替代方案、班次选择等决策变量，以及产能、模具匹配等约束条件；进而采用混合整数规划（MILP）与启发式算法（如数学规划启发式、元启发式等）进行多目标优化，权衡成本最小化、准时交付与资源限制等多重目标。通过动态决策产能扩容、安全库存设置及替代生产触发机制，有效应对生产过程中的不确定性，最终在复杂多级生产场景中实现成本效率与交付效率的高度协同。

3.1.2 批量计划问题分类

本节将系统解析批量计划问题的理论框架与实践脉络。首先建立基础问题建模的数学描述，明确核心变量、约束与目标；进而梳理经典问题分类体系（如CLSP、MLCLSP等），解析其理论特征与求解难度[98][99]。

3.1.2.1 批量计划问题模型

实际批量计划问题的变种繁杂，从制造系统底层因素来看，主要涉及产品及其结构、机组、时间的关联关系。本小节针对基础的CLSP问题的数学模型进行描述，考虑多级生产模式、多种产品、产能扩容以及多个生产时间段等，通过核心的变量和约束帮助读者理解问题核心背景。

（一）集合及索引

j 物品 （如 终端产品、中间产品）,j=1, ..., J

m 资源 （如 人力、机器、产线）,m=1, ..., M

t 周期，t=1, ..., T

S_j BOM中物品i的后续产品

（二）模型参数

a_{mj} 生产一个物品j所需资源m的量

B_{jt} 一个大数，不会限制物品 j 在周期 t内的批量大小

C_{mt} 周期t内资源m的可用产能

h_j 在一个周期内持有一个单位物品j的持有成本

oc_{mt} 在周期t内多使用一个单位资源m的超期成本

P_{jt} 物品j在周期t内的需求量

r_{jk}^{d} 生产一个后序物品k，如要当前物品j的数量

sc_j 物品j的setup成本

st_{jm} 物品j占用资源m的setup时间

（三）决策变量

I_{jt} 周期t期末物品j的库存量

O_{mt} 周期t中资源m的超期量

X_{jt} 周期t中物品j的生产量

Y_{jt} setup变量（如果物品j在t时刻生产，则为1，否则为 0）

（四）问题建模

$$Min. \quad \sum_{j=1}^{J}\sum_{t=1}^{T} h_j \cdot I_{jt} + \sum_{j=1}^{J}\sum_{t=1}^{T} sc_j \cdot Y_{jt} + \sum_{m=1}^{M}\sum_{t=1}^{T} oc_{mt} \cdot O_{mt}$$

Subject to

$$I_{jt-1} + X_{jt} = P_{jt} + \sum_{k \in S_j} r_{jk}^{d} \cdot X_{kt} + I_{jt} \qquad \forall j = 1,\ldots,J, t = 1,\ldots,T$$

$$\sum_{j=1}^{J} a_{mj} \cdot X_{jt} + \sum_{j=1}^{J} st_{jm} \cdot Y_{jt} \le C_{mt} + O_{mt} \qquad \forall m = 1,\ldots,M, t = 1,\ldots,T$$

$$X_{jt} \le B_{jt} \cdot Y_{jt} \qquad \forall j = 1,\ldots,J, t = 1,\ldots,T$$

$$I_{jt} \ge 0 \qquad \forall j = 1,\ldots,J, t = 1,\ldots,T$$

$$O_{mt} \ge 0 \qquad \forall m = 1,\ldots,M, t = 1,\ldots,T$$

$$X_{jt} \ge 0 \qquad \forall j = 1,\ldots,J, t = 1,\ldots,T$$

$$Y_{jt} \in \{0,1\} \qquad \forall j = 1,\ldots,J, t = 1,\ldots,T$$

3.1.2.2 问题分类

在经典的批量计划问题学术研究中，按问题结构（单级/多级）、约束类型（资源/时间/库存）、交付限制（提前期、时间窗）等进行分类。经典模型如单级产能受限批量计划问题（CLSP）与多级扩展（MLCLSP）构成理论基础，进一步通过引入多资源、动态需求、替代生产等特性逐步拓展问题复杂度。此类分类体系为算法设计与理论分析提供了框架，但需结合实际生产场景的差异化需求进行适配。具体问题分类表格详见表3.1。

表3.1 经典批量计划问题分类

问题名称	英文全称	问题描述
CLSP	Capacitated lot-sizing problem	产能受限的批量生产计划
CLSP-MP	Multi-plant capacitated lot-sizing problem	多工厂协作 CLSP
CLSP-PM	Capacitated lot-sizing problem with parallel machines	多台机器并行的批量生产计划，增大一个维度
MLCLSP	Multi-level capacitated lot-sizing problem	考虑多层 BOM

CLSP-OV	Capacitated lot-sizing problem allowing overtime	允许超出产能的 CLSP
CLSP-BO	Capacitated lot-sizing problem with backorders	允许未满足需求的 CLSP
CLSP-II	Capacitated lot-sizing problem with initial inventory	考虑初始库存的 CLSP
CLSP-MINQ	Capacitated lot-sizing problem with minimum production quantity	考虑最小产量的 CLSP
CLSP-TW	Capacitated lot-sizing problem with time windows	指定需求生产期次范围的 CLSP
CLSP-ST	Capacitated lot-sizing problem with setup time	考虑换型时间的 CLSP，因为常用的 FIX 方法容易超出容量约束而增加了问题难度
CLSPL	Capacitated lot-sizing problem with linked lot-sizes (or setup carryovers)	需要决定每期始末产品类型（跨期生产产品可省略二次换型），引入次序决策
MLCLSPL	Multi-level capacitated lot-sizing problem with linked lot-sizes (or setup carryovers)	多层 CLSPL
CLSPL-LD	Capacitated lot-sizing problem with linked lot-sizes considering lead time	考虑多层 BOM 时，引入产品生产提前期
CLSP-SD	Capacitated lot-sizing problem with sequence dependency setup time	考虑换型时间且换型时间和次序相关的 CLSP，引入次序决策，为 scheduling 问题，通常用启发式解决
CLSP-SUB	Capacitated lot-sizing problem with substitution	考虑多层 BOM 时，引入物料替代关系

3.1.3 批量计划求解器技术架构

批量计划求解器的技术架构如图3.1所示，主要包含底层天筹AI求解器，批量计划的模型、算法、工具库，以及通用接口和配置层。此外针对不同行业以预制模板的形式进行初始化配置，增强行业适配性和易用性。

图3.1 批量计划求解器技术架构图

1) 预置行业模板：系统预设了多个行业模板，如轮胎、烟草、医药等，每个模板针对特定行业特点，如轮胎行业的“大规模同质机组”和医药行业的“药品标准限制”和“按批MTS模式”。这些模板为不同行业提供了行业解决方案的引擎组件；

2) 引擎配置层：该层定义了优化目标、生产约束和算法，是系统的核心配置部分。优化目标包括生产成本、资源成本、库存成本和延期成本等；生产约束涵盖产能、资源、库存、人力等多方面；算法则包括联合迭代、数学规划、邻域搜索等，确保系统能够根据具体需求进行灵活配置；

3) 模型库与算法库：模型库包含优化目标和生产约束的详细分类，算法库则提供了多种优化算法，如联合迭代、数学规划、邻域搜索等，支持复杂生产问题的求解；

4) 辅助工具：包括输入检测、不可行分析和结果解读等工具，帮助用户进行数据验证和结果分析，提升系统的可靠性和实用性；

5) 天筹AI求解器：作为底层核心，包含数学规划求解器、启发式算法库和黑箱优化算法库，为系统提供强大的计算能力和优化能力，确保生产计划的高效性和准确性。

3.1.4 批量计划求解器求解流程

批量计划求解器的具体求解流程如图3.2所示，主要包含对结构化数据的读取和问题解析，使用引擎内置的建模方法，根据不同的问题类型生成对应的批量计划问题建模，最后通过求解技术对问题进行求解和结果输出。

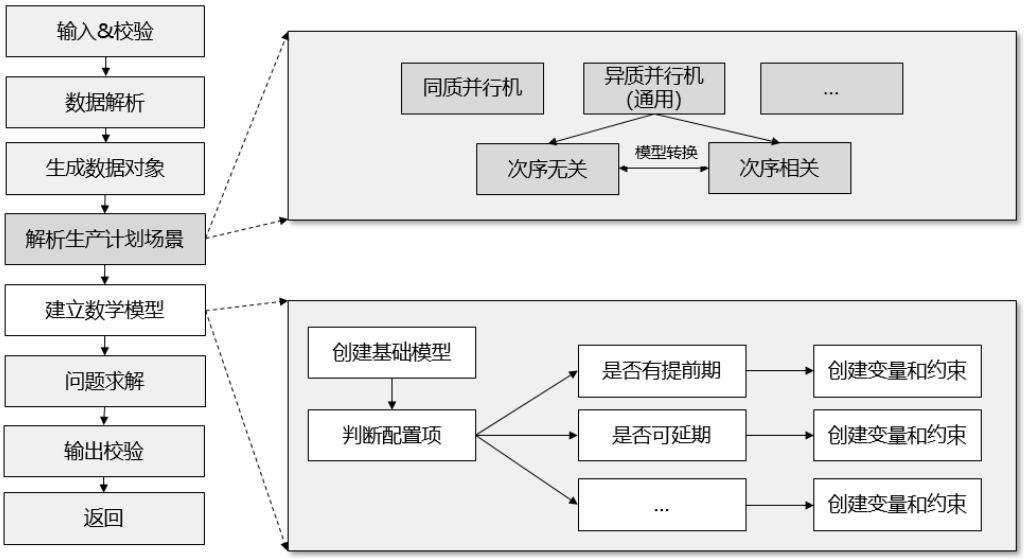

图3.2 批量计划求解器求解流程图

整体流程解读如下：

1) 输入读取：算法首先通过 JSON 文件读取多类输入信息，其中包括产品、BOM 结构、订单、机器、模具、库存、班次以及成本等参数，并将其解析为对应的数据结构进行存储；

2) 数据解析和参数配置：需要根据不同的数据类型选择不同的策略，可配置黑箱优化等技术对算法参数进行进一步优化；

3) 模型求解阶段由初始化构造与局部搜索两部分组成。初始化部分结合启发式算法、简化模型、松弛和修复约束等多种策略构造初始可行解；

4) 优化阶段：采用局部搜索和迭代策略一步优化，其中局部搜索通过邻域评估技术，从机器维度、班次维度、序列维度替代维度、时间维度出发，以邻域动作优化解；

5) 结果输出：达到运行时间或者最优解后通过JSON格式输出结果。

3.1.5 批量计划求解器功能特性

在工业实践中，批量计划问题需根据企业生产模式（离散/流程制造）、资源特性（设备/人力/模具）、业务规则（保质期/切换成本）等约束进行特性组合以描述复杂场景问题，通常涉及到的约束类型比经典问题更多且组合复杂。因此基于行业应用实践对通用特性进行抽象提炼，总结出以下通用应用特性通分类，通过集成场景特性模板，执行“建模—>求解—>输出”的功能，以场景化接口降低建模门槛。具体的场景化的功能特性分类详见表3.2。

表3.2 批量计划求解器特性列表

功能类型	功能点	功能描述
模具	单模具/共享模具	生产过程中只涉及同一种模具/多种产品共享同一种模具
并行机	同质并行机	同一产品可同时在多台机器上进行生产，且同一产品在不同机器上的生产效率及能力无差异
	异质并行机	同一产品可同时在多台机器上进行生产，且同一产品在不同机器上的生产效率及能力不同
安全库存	安全库存	为应对需求或供应的不确定性对产品库存设定目标，在满足需求的情况外尽量接近该目标
最大库存	最大库存	受场地限制，单产品库存不超过其最大库存
多层 BOM	多层 BOM	产品需要经过多道工序后完成，上层产品的生产需要消耗对应下层产品
产品提前期	产品提前期	在多层 BOM 中由于运输、工艺要求等原因下层产品经过一定时间后（提前期）后才可以用来生产上层产品
可替代	生产替代	生产过程中的半成品可以用别的产品替代
	替代优先级/后继料	不同产品之间的替代有优先级区别，老版本物料尽量优先消耗
	过度替代	用替代产品来满足需求的量不可超过实际发生的加工或交付需求
过期	过期	产品的库存只可存放一段时间（有效期），过期后无法使用
需求延期	需求延期	需求可被延期满足（在后续时间生产），但有延期成本
缺货	缺货(lost sale)	需求如果未被及时满足则直接缺货，后续不再重新生产，但有缺货成本
需求时间窗	需求时间窗	产品的需求必须在特定时间内完成
换型结转	换型结转（carry over）	产品在下一周期初可继续生产时，不需要考虑 setup 时间和成本
换模/换型时间	次序相关	产品之间的生产切换时间和成本与产品切换顺序相关
换模/换型次数限制	换模/换型次数限制	生产过程中产品之间的生产切换产生的换模/换型次数不可超过上限
产能扩容	产能扩容	生产过程中的产能不足时允许扩充

模具扩容	模具扩容	模具不足时，可考虑扩容
生产量	最小最大批量	生产时单次生产批量有上限/下限限制
	最小最大产量	生产时单次生产量有上限/下限限制
	配对生产	生产不同产品时必须按指定比例生产（生产 m 个 A 产品必须生产 n 件 B 产品）
加工冻结期	加工冻结期	由于机器维护或试运行、原料未到位等原因，开工后数天内不允许生产
行业通用	产能平衡	不同产线集合的产能使用率差值要满足给定的产能差异限制
	多品类聚合批量限制	支持多个品类在多个机台上聚合后满足批量限制需求
	多品类聚合产量限制	支持多个品类在多个机台上聚合后满足产量限制需求
	环境能耗	多个机台集合的最大产能使用时间关联该组机器环境能耗消耗
	一对多模具	辅助机器可一对多连接生产机器，但只能用于相同产品（例如：烟草行业喂丝机）
	可换型不可生产产能	支持设置只能用于启动换型不能进行生产的产能
	机器模具/人力约束	支持设置机器开机需要的模具或人力资源，只与机器关联，与所生产产品无关
	整数性产量	输出产量数值需满足整数性要求
	初始机器启动产品	设置机器初始启动生产产品状态

3.1.6 批量计划求解器关键技术

在MLCLSP问题中，决策变量通常是离散的，如生产顺序、启动换模等。这导致问题无法直接应用传统的连续优化方法进行求解。并且由于这类问题通常是NP难问题，采用离散优化方法或混合整数规划方法直接进行求解同样难以求解。因此，本章节将重点介绍高效求解批量计划类问题的算法架构以及常见方法分类，对其中关键方法的详细步骤进行总结。

3.1.6.1 批量计划求解算法架构

批量计划引擎求解模块的详细架构如图3.3所示，该模块包含了初始解算法模块、优化模块以及启发式算法路线，以下将对具体模块内容进行说明。

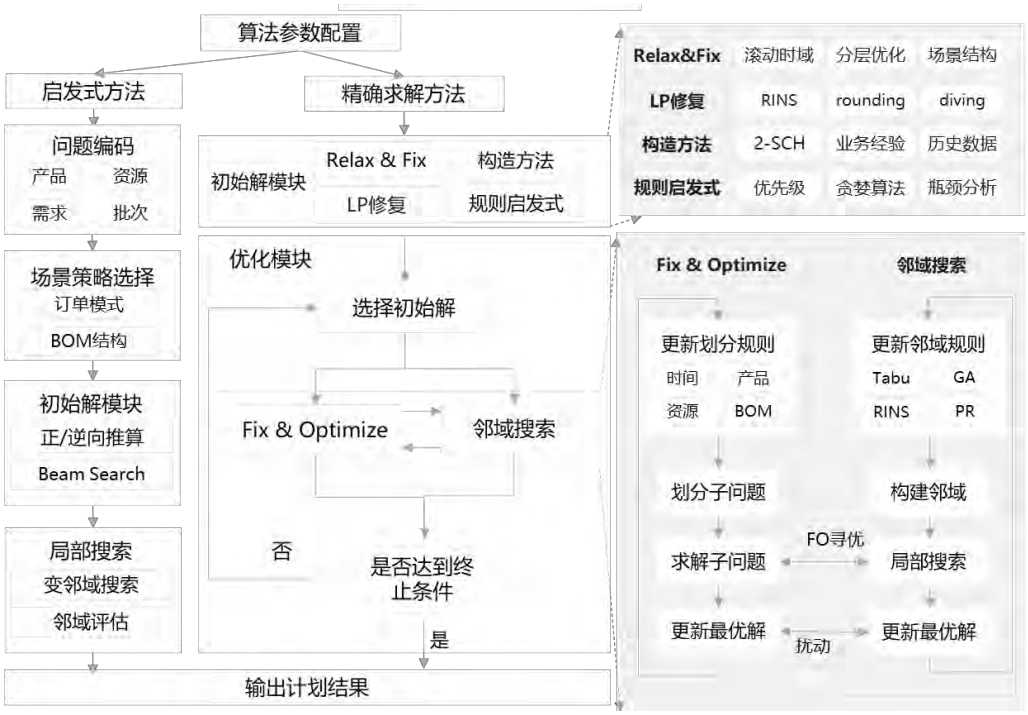

图3.3 批量计划求解器算法架构图

（一）初始解模块

1) 构造方法：通过多种经验性方法或历史数据实现快速构造初始解，并使用简单的前移/后移方法进行优化；

2) LP+修整：利用松弛初始解和修整策略快速构造可行解；

3) Relax & Fix：滚动时域等方法结合MIP求解器获得优质可行解；

4) 规则启发式：基于订单/需求/工序/资源等优先级进行排产，也可以结合瓶颈资源或工序分析进一步迭代优化。

（二）优化模块

1) Fix & Optimize：通过时间、产品、资源等维度划分子问题缩小问题规模，利用MIP求解器高效迭代求解；可引入扰动算子；

2) 邻域搜索：结合多种迭代邻域和变邻域规则提升邻域结构多样性，高效迭代跳出局部最优；局部搜索利用FO算法寻优。

（三）纯启发式求解算法路线

1) 问题编码：对产品、资源、需求、库存、辅助模具以及计划批次等场景元素进行对象建模，利用高效编码方式转换计划结果；

2) 初始解构造方法：初始化包含了正向推算、逆向推算、Beam Search等多种

策略，旨在获得一个较优的可行解；

3) 改进启发式：利用局部搜索和树搜索进一步优化，其中局部搜索通过邻域评估技术，从机器维度、班次维度、序列维度替代维度、时间维度出发，以邻域动作优化解。

3.1.6.2 批量计划求解算法分类

求解CLSP问题的主要方法[98]可以包括：(1) 精确算法，如拉格朗日松弛(LR)、分支-定价(B&P) 、列生成(CG)、分支-割平面(B&C)、Benders分解(BD)等；(2) 数学启发式，如固定-元整(Rounding)、松弛-固定(R&F)、固定-优化(F&O)等；(3) 元启发式，如局部搜索(LS)、变邻域搜索(VNS)、禁忌搜索(TS)、遗传算法(GA)等；(4) 分层求解；(5) 贪婪启发式，具体如下表（表3.3）所示。

表3.3 批量计划求解方法分类

方法	特点	优点	缺点	求解问题
精确算法 1）拉格朗日松弛(LR) 2）分支-定价(B&P,CG) 3）分支-割平面(B&C) 4）Benders 分解(BD)	1）基于模型结构分解 2）子问题可精确求解 3）主子问题迭代优化 对偶理论验证最优性	1）有数学理论支撑(dual) 2）解的质量有保障(gap) 3）调用已有算法(DP)	1）适合小规模问题T=[3,30],I=[5,15],M=[3,5] 2）实施复杂度高 3）依赖模型结构 4）可扩展性低	1）CLSP 2）CLSP-SQ
数学启发式 1）固定-元整(Rounding) 2）松弛-固定(R&F) 3）固定-优化(F&O)	1）松弛或分解原模型 2）子模型调用求解器 3）利用部分解信息 4）缩小模型规模	1）求解器赋能，提效增速 2）通用性较强	定制化内容需后处理	1）CLSP 2）CLSP-SQ 3）CLSP-C
元启发式 1）局部搜索(LS) 2）变邻域搜索(VNS) 3）禁忌搜索(TS) 4）遗传算法(GA)	1）通用框架 2）通过调参提升效果	1）实现简单 2）灵活性较高 3）求解速度快 4）调参获得高质量解	1）针对不同问题参数敏感 2）部分算子设计依赖问题	1）CLSP 2）CLSP-SQ 3）CLSP-C
贪婪启发式 根据问题结构定制化	融合行业知识，优先指标，保证可行性	1）融合行业知识	定制化开发，通用性差	1）CLSP 2）CLSP-SQ

		2）保证可行性和效果		
分层求解 分解问题，逐层优化	将问题分解为多个决策过程，分层求解	1）简化问题，加速求解 2）多模块相对独立场景	各模块相对独立，信息交互和更新要求高	1）CLSP 2）CLSP-C

（一）精确算法

精确算法包括拉格朗日松弛法[100]、分支定价算法以及分支割平面算法。其中分支定价算法一般采用Dantzig-Wolfe从时间、产线或产品维度进行分解原问题，其子问题可以结合单纯形法、次梯度法或RINS、Selective Diving和Successive Rounding等启发式加速求解，最终根据分支树保证解的最优性。分支割平面算法则根据问题模型添加若干有效不等式，尽可能描述解空间的凸包或facet有效提高子问题求解效率。拉格朗日松弛法则通过松弛产能或换模相关等复杂约束加到目标函数上进行惩罚，通过不断更新拉格朗日乘子得到持续被优化的近似解，其子问题通常被转换为经典问题后使用动态规划快速求解，由于其通常能够获得比线性松弛更优下界，因此通过可以获得更有可行解。

（二）数学启发式算法

数学启发式[42][101][102]一般较为通用，其核心思想是固定大部分变量再优化剩余变量，通过减小问题规模来提高子问题求解效率。最为常见的是固定松弛与固定优化两种启发式算法，其关键步骤在于如何对变量集合进行合理划分[103]，使得子问题之间耦合程度较少。固定松弛算法一般用于构建初始可行解，其每次根据当前最优解对前几次已经优化过的变量进行固定，而当前选择的一个或多个集合变量进行优化，剩余0-1变量则进行松弛来构建子问题，通过不断迭代求解直至所有变量都至少被优化一次，从而得到最终的高质量可行解。固定优化算法则无法构建可行解，主要用于优化可行解，与固定松弛算法不同点在于松弛的变量均按照最好可行解进行固定，因此每次能够保证当前解至少不会差于当前最好解。这种提前对变量分解并迭代优化的过程也被称为滚动时域法，即将每一个变量集合看做一个时域并按顺序进行遍历。

固定优化方法较为通用，每次优化的变量不一定是互斥不相同的，常见的方法还有根据上下界差距大小、0-1变量被优化的次数、以及问题模型特征选择待优化变量集合，但核心思想均通过减小问题规模来提高求解效率。

（三）元启发式算法

元启发式算法相较于数学启发式算法更为通用，一般使用禁忌搜索算法、遗传算法[104]或变邻域搜索算法等。其同样会针对模型设计出例如生产变量移动方法等邻域进行搜索，或者通过遗传算法将可行解进行编码转换再进行自定义交叉变化从而跳出局部最优。其通常求解速度较快，且实现较为迅速但针对不同问题参数过于敏感，为了获得高质量可行解，通常需要进行较多调试。

（四）贪婪启发式算法

贪婪启发式是一种高度定制化的近似优化策略，其核心在于深入挖掘问题的内在结构与行业知识，构建符合业务逻辑的逐步求解规则。例如，在生产计划场景中，可基于“优先排产交货期紧迫的订单”或“优先选用成本更低的机器”等规则进行决策。该方法能够高效生成可行解，并因融入领域经验而具备良好的实用质量。然而，其启发式规则通常针对特定场景设计，通用性有限，且无法保证解的最优性，甚至可能因规则设计不当而陷入明显的次优状态。因此，贪婪启发式常作为复杂优化问题的快速启动策略，或嵌入其他算法中辅助局部决策，典型应用包括带设置时间的单级批量计划问题（CLSP-SQ）等场景。

（五）分层求解算法

分层求解是一种通过分解与协调机制来应对复杂决策问题的有效方法。该方法将大规模、高耦合的复杂系统（如多级生产计划）依据决策层级或功能模块拆分为若干相对简单的子问题，随后按特定顺序（例如自上而下，从战略层到战术层）或迭代方式进行逐层求解。举例来说，可先制定长期产能规划，再基于该规划结果进行具体排产调度。这种策略的优势在于显著降低了单次求解的复杂度，提升了计算效率，并允许各模块独立开发与优化。然而，其核心挑战在于如何有效协调各层之间的信息传递与反馈，避免因下层结果与上层假设冲突而导致整体方案失效。因此，设计合理的反馈机制成为分层求解成功实施的关键。该方法在多级有限能力批量计划问题（Multi-Level Capacitated Lot Sizing Problem，MLCLSP）等具有层次化结构的复杂场景中具有广泛应用。

3.1.6.3 松弛-固定初始解算法

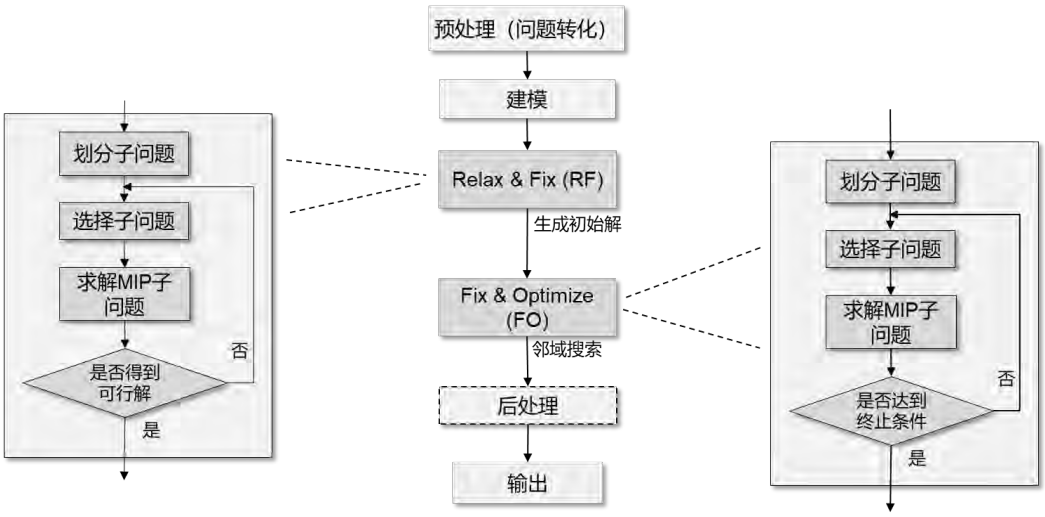

图3.4 批量计划RF&FO数学启发式框架

松弛-固定算法（Relax-and-Fix，RF）是一种用于解决组合优化问题的启发式算法[105]，通常用于整数规划问题，它常与固定-优化算法（Fix-and-Optimize，FO）联合使用，整体流程见图3.4。该算法的基本思想是通过在原问题中固定一些变量，然后求解相应的松弛问题，然后在松弛问题的解的基础上更新原问题中的变量；这个过程不断迭代，直到满足终止条件。这种方法的优点在于它结合了整数规划和松弛问题的求解过程，通过在每一步中逐渐锁定变量的值，可以有效地搜索整数解的空间。然而因为算法可能陷入局部最优解，这并不保证找到全局最优解。RF整体算法示意图如图3.5所示，黑色为固定变量集合，灰色为优化变量集合，白色为松弛变量集合，从上到下逐个根据移动步长移动优化时间窗来构建不同子问题。

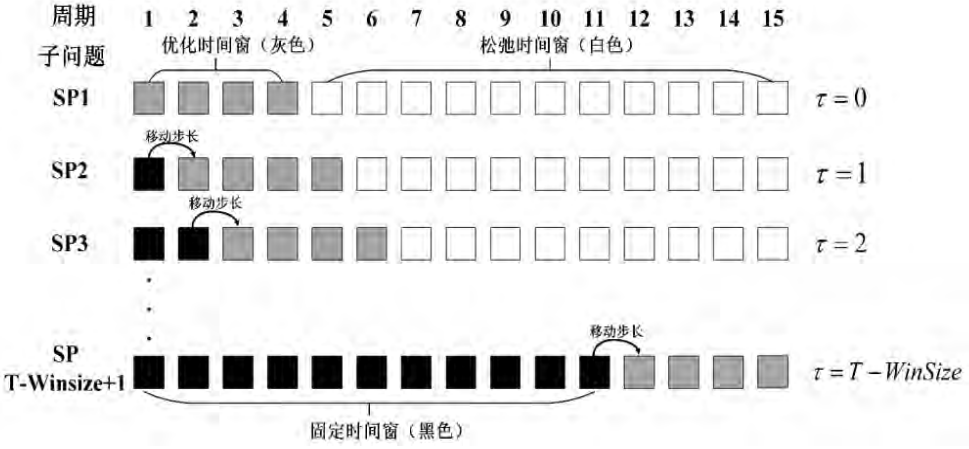

图3.5 批量计划RF算法原理示意图

RF算法流程如下：

步骤1：初始化参数，包含线程数、变量集合数、划分变量组数量，以及移动步长、优化窗大小等。

步骤2：设置子问题求解时间和求解方式，通过步长和时间窗参数、问题规模等动态调整。

步骤3：构建子问题：根据当前线程自适应参数，对指定变量集合进行固定、优化、松弛等操作。这几个集合分别定义如下：

1) 优化变量：优化时间窗内的所有整数变量；

2) 固定变量：使用当前最优可行解，对已搜索范围内整数变量进行固定（不包含优化时间窗内的变量）；

3) 松弛变量：对位搜索到的变量范围内的整数变量进行松弛；

步骤4：对松弛、固定、恢复后的问题利用求解器进行求解。

步骤5：终止条件判断，若达到则退出，若未达到则进入步骤6：

1) 达到最大求解时间；

2) 松弛变量全部为整数变量。

步骤6：更新算法参数并返回步骤3，根据移动步长大小及优化窗大小更新优化窗范围区间。

3.1.6.4 固定-优化优化算法

固定-优化算法（Fix-and-Optimize，FO）算法是一种通常用于组合优化问题的优化算法[106][107]，该算法的基本思想是在每次迭代中，先固定一部分变量，然后对固定后的问题进行优化。这一过程反复进行，直到找到满足特定条件的解，或者达到迭代次数的上限。其算法的优势在于它在每次迭代中可以通过固定一部分变量来降低问题的复杂度，从而更容易找到局部最优解。然后，通过重新引入固定的变量，算法可以探索更广泛的解空间，寻找全局最优解。这种算法适用于许多组合优化问题，例如车辆路径问题、调度问题等。

但FO仅可以用于优化可行解，其本身无法直接生成可行解，因此必须以一个初始可行解作为起始，常见方法大多以与RF算法为其提供可行解，两者在划分整数变量集合方法一致，仅有部分变量处理不同，两方法结合在一般规模的MIP模型

都有不错的算法效果。

Fix & Optimize算法的变量划分方式与RF算法类似，在预处理阶段需要采用相同的方法对模型变量进行划分和自适应参数分配。二者的关键区别在于构建子问题时的变量处理策略：Fix & Optimize在确定优化变量的范围后，会将剩余变量固定为当前最优可行解的值，而不是采用松弛方式处理。这一策略确保了每个子问题求解后得到的解均为原问题的可行解，从而能够在每一步有效优化当前最优可行解，同时保证了解的可行性和收敛效率。

FO算法具体实现过程可以分为以下几步：

步骤1：使用初始解模块提供初始可行解，作为FO算法的初始可行解。

步骤2：初始化各线程参数，同RF算法。

步骤3：构建子问题。

优化变量：优化窗$[ed_{n-1}+1, ed_n]$内的整数变量。

固定变量：除优化窗$[st_{n-1}, st_n-1]$范围内的所有整数变量。

步骤4：求解子问题。

步骤5：终止条件判断，若满足则退出，否则进入到步骤6。

步骤6：算法参数更新（同RF），根据移动步长大小及优化窗大小更新优化窗范围区间。并根据当前算法实际求解时间调整优化窗大小。返回步骤3。

3.1.6.5 启发式固定初始解

针对生产计划问题的单机问题，论文中常用的初始解生成方式是，对原问题进行数学建模生成MIP模型，并假设在机器上所有产品的启动变量均为1，固定这些整数变量后求解LP问题，得到原问题一个较差但保证可行性的初始解（问题中额外产能无限，对机器上的换型次数没有限制），再使用优化模块进行邻域搜索。然而，当约束更多、场景更复杂时，需要设计定制的启发式方式进行适配，以达到相似的效果。

（一）异质并行机场景

在异质并行机场景中一个产品可以在多个机器上进行生产，如果同时固定所有启动变量，即令每个产品在所有时间在所有可用机器的启动变量为1很可能导致初始解的质量差，成本也会较高，后续优化的难度较大，拓展性不强。因此对原固

定策略进行改进。主要变化为，在生成初始解时，每个产品依据一定的方法只选择一台可用机器作为产品的默认加工机器，并在整个优化周期中保持不变，将该产品在该机器上所有时间的启动变量固定为1，在其它可生产机器的启动变量固定为0。

以下为主要的生成步骤：

1) 将所有产品在可加工机器上的启动变量预先设置为关闭；

2) 将所有只能在一台机器上生产的产品进行机器分配固定，根据每个分配的产品的平均需求量计算预期的使用产能，更新每台机器的估计剩余产能；

3) 按照真实BOM展开后的需求量从大到小对产品进行排序；

4) 对排序后的产品依次进行遍历，选择每个产品可加工机器中剩余产能最多的机器作为分配机器，如果剩余产能可以启动生产该产品，则分配该产品到当前机器；如果剩余产能不足以启动该产品生产，则放弃该产品的分配；

5) 分配结束后，按照分配机器结果将产品的启动变量设置为开启状态。

为避免初始解不可行情况，如果该预置分配方案没有得到可行解，将采用其他备用（如直接使用求解器）的方法进行求解尝试。

（二）同质并行机场景

在同质并行机场景下，构造启发式算法时需为每个产品在每个时间周期分配所使用的机器种类及数量，以确定MIP模型中的整数变量取值。在此基础上，调用MIP求解器进一步求解LP子问题，以确定产量、产能分配等连续变量的值。然而，由于存在如下约束：第t时刻用于生产产品i的机器总数必须等于第t+1时刻继续生产产品i及由产品i切换至产品j所需的总机器数，这一约束在构造可行解时较为复杂。为简化初始解的构造过程，我们假设在整个生产周期内，每个产品所使用的机器种类及数量不随时间变化。这一简化有助于快速生成初始可行解，并为后续优化过程提供基础。

同质并行机的构造启发式算法构造初始解的流程主要分为三部分：

1) 为只有一种可选机器的产品分配机器及数量；

2) 为剩余产品依次选择总产能最大的机器进行分配种类及数量；

3) 固定机器分配的整数变量，求解LP问题。

在步骤2)中可采用不同的排序方法为剩余产品排序。

3.1.6.6 路径重连

路径重连（Path-relinking，PR）是一种元启发式搜索策略[108][109]，常用于组合优化问题（如生产批量计划），通过在解空间中构建路径连接两个解（如局部最优解或高质量解），以探索潜在的更优解。其核心思想是通过系统性地扰动和修复解，在解空间中生成新解，从而跳出局部最优。

步骤 1：初始化解集，使用 RFO 获取三个高质量解。

1) 第一个解为RF在$\theta-3$次迭代对应解（后续时域变量松弛可能为不可行解），其中θ为RF总迭代次数；

2) 第二个解初始化为RF对应可行解并由PR持续进行更新；

3) 第三个解为RFO最终可行解作为引导解，并将持续被全局最优解所更新。

$$(x^{\mathcal{E}(1)}, y^{\mathcal{E}(1))}, z^{\mathcal{E}(1)}) \leftarrow (x^{RF}, y^{RF}, z^{RF});$$
$$(x^{\mathcal{E}(2)}, y^{\mathcal{E}(2))}, z^{\mathcal{E}(2)}) \leftarrow (x^{RF^*}, y^{RF^*}, z^{RF^*});$$
$$(x^{\mathcal{E}(3)}, y^{\mathcal{E}(3))}, z^{\mathcal{E}(3)}) \leftarrow (x^{FO}, y^{FO}, z^{FO});$$

步骤 2：初始化其他额外参数。

1) 给定限制时间$\xi \leq p$：其中ξ之后的所有时域都不会被约束固定；

2) 有效减少setup物品集合：统计setup时间/费用较小物品集合，防止被固定，通常从所有物品中选择Setup费用较小10%物品。

步骤3：PR初始化。

1) 目标值：当前最优RF+PR结果， $Z^{PR} \leftarrow Z^{\varepsilon(2)}$；

2) 求解时间：TimeLimit-PR/2。

步骤4：求解子问题（M1）。

1) 构造子问题：基于三个高质量解，固定无效setup变量，并且保证限制时间之后不允许固定，并且不能从能够降低setup费用中的物品中选择；

2) 求解器求解M1问题，获取M1可行解$(x^{(M1)}, y^{(M1)}, z^{(M1)})$；

3) 更新PR解：如果M1解更优则更新，$(x^{(\varepsilon 2)}, y^{(\varepsilon 2)}, z^{(\varepsilon 2)}) \leftarrow (x^{(M1)}, y^{(M1)}, z^{(M1)})$；

4) 更新FO解：若$\varepsilon 2$解更优，则进一步基于当前解进行FO，求解时间为剩余PR限制时间。

$$(x^{(\varepsilon 3)}, y^{(\varepsilon 3)}, z^{(\varepsilon 3)}) \leftarrow \boldsymbol{FO}(x^{(\varepsilon 2)}, y^{(\varepsilon 2)}, z^{(\varepsilon 2)})$$

步骤 5：统计最优解。

1) 判断终止条件：若尚有剩余求解时间，且当前PR解与FO解不一致，则重复步骤4；

2) 若达到终止条件，首先判断两个解的可行性，若均不可行则输出错误求解信息。否则，选择更优解作为最终解。

3.1.6.7 两阶段构造启发式

（一）算法描述

两阶段构造启发式算法（Two-step construction heuristic，2-SCH） 是一种用于快速构造生产计划问题的构造启发式算法[110]。该算法的基本思想是在第一步中对客户的订单进行排序，并在第二步中迭代地将其添加到一个初步的生产计划中，并依照一定规则确定生产计划各周期决策变量的值。这个过程不断迭代，直到全部订单都添加进生产计划中。这种方法的优点在于每次仅在当前生产计划基础上添加一个客户订单，使各类问题可以轻松、快速地解决。然而，这并不保证找到全局最优解，因为算法可能陷入局部最优解。

（二）算法实现过程

步骤1：初始化：对生产计划初始决策变量值进行设定。

1) 各产品各周期生产批量大小：设定为0；

2) 各产品各周期设置生产状态：设定为0；

3) 各机器各周期产能可用量：额定产能，增补量：设定为0；

4) 各产品初始库存量扣除已知需求。

步骤2：选取订单集合排序方式。

2-SCH 算法需要选取一种排序方式对订单集合进行排序，其中订单排序方式包括以下几种方式：

1) 按订单量排序：从订单集合中获取所有订单对应订单量，根据订单量从大到小进行排序；

2) 按时间维度排序：从订单集合中获取所有订单对应时间周期，根据时间周期从前到后进行排序；

3) 按产品维度排序：从订单集合中获取所有订单对应产品序号，根据产品序号从小到大进行排序；

4) 按订单量与时间维度排序：从订单集合中获取所有订单对应订单量和对应时间周期，根据订单量从大到小进行排序；当订单量相同时，根据时间周期从前到后进行排序；

5) 按订单量与产品维度排序：从订单集合中获取所有订单对应订单量和对应产品序号，根据订单量从大到小进行排序；当订单量相同时，根据产品序号从小到大进行排序；

6) 按时间维度与产品维度排序：从订单集合中获取所有订单对应时间周期和对应产品序号，根据时间周期从前到后进行排序；当订单时间周期相同时，根据产品序号从小到大进行排序；

7) 按时间维度与订单量排序：从订单集合中获取所有订单对应时间周期和订单量，根据时间周期从前到后进行排序；当订单时间周期相同时，根据订单量从大到小进行排序；

8) 按产品维度与订单量排序：从订单集合中获取所有订单对应产品序号和订单量，根据产品序号从小到大进行排序；当订单产品序号相同时，根据订单量从大到小进行排序；

9) 按产品维度与时间维度排序：从订单集合中获取所有订单对应产品序号和时间周期，根据产品序号从小到大进行排序；当订单产品序号相同时，根据时间周期从前到后进行排序

步骤3：添加订单后修改决策变量值。

依照订单集合排序，依次将订单集合中订单加入生产计划，根据模型约束更新决策变量值，迭代过程包括如下四个步骤：

1) 添加订单（需求）；

2) 根据当前订单（需求）及可用产能，计算可用的生产策略：

场景1 (use inventory): 利用当前周期的库存满足需求，并在下个周期创建新的需求

场景2 (sufficient capacity): 当前周期的产能充足，尝试extend(在之前已生产的周期利用剩余产能生产)，new(在前一个产能充足且未生产的周期创建生产)

场景3 (insufficient capacity in t, while sufficient capacity in 1~t): 当前周期的产能不足充足，尝试扩展extend(在之前已生产的周期利用剩余产能生产)

场景4 (insufficient capacity in 1~t): 利用额外产能, 在当期生产并利用额外产能进行生产；

3) 根据当前场景进入到不同的case，在对应的case下选择成本最小的一种策略更新生产计划，即生产状态、生产量、可用产能、额外产能、库存量、生产成本。

步骤4：对生产计划进行优化，将生产量左移（left shift）或右移 (right shift)，尝试减少生产成本。

1) 左移: 将t周期之后的第一个生产量左移到当前周期（保证当前可用产能足以满足该生产量），检查该移动增加的库存成本与减少的setup成本，是否会使总成本减小；

2) 右移：将t周期之前有库存量且有生产计划的产品生产右移至当前周期，检查该移动减少的库存成本和增加的生产成本（若新创建setup），是否会使总配成减小。

步骤5：判断如下终止条件，若达到则退出，否则返回步骤3。所有需求全部添加入生产计划时，迭代终止。

3.2 生产调度求解技术

3.2.1 生产调度应用背景

生产调度问题作为运筹学中的经典组合优化问题[111][112]，聚焦于在有限时间和多重资源约束下，对生产任务进行高效排序与资源分配，以实现总完成时间最小化、设备利用率最大化等关键生产指标。该问题通常涉及多个工件在多台机器上按特定工艺路线加工，需满足工序顺序约束和机器独占性约束，具有显著的NP-hard特性。其类型包括单机调度、并行机调度、流水车间调度、作业车间调度以及更贴近实际生产的柔性作业车间调度等[113][114][115]。传统运筹学方法如数学规划和约束规划虽然能保证最优解，但计算效率有限，因此启发式算法成为解决大规模实际问题的有效手段。该问题的优化直接关系到制造业的生产效率、资源利用率和运营成本，并在绿色制造与智能制造背景下逐步扩展至能耗优化与多目标协同优化等方向。

3.2.2 生产调度问题分类

根据具体调度场景选择合适的问题模型，常见类型包括不限于：

1) 作业车间调度（JSP）：每个工件有特定加工顺序，工序需在指定机器上加工；

2) 流水车间调度（FSP）：所有工件按相同顺序经过一系列机器；

3) 柔性车间调度（FJSP）：工序可在多台可选机器上加工，增加资源灵活性；

4) 混合流水车间调度（HFSP）：结合流水车间与并行机特点，某些阶段存在多台并行机器。

常见约束条件包括不限于以下类型：

1) 工序顺序约束：工件工序必须按预定顺序加工；

2) 机器独占性：一台机器同一时间只能加工一个工序；

3) 资源限制：如模具、人员等辅助资源约束；

4) 时间约束：如交货期、准备时间、传输时间等。

3.2.3 生产调度求解器技术架构

生产排成引擎架构可分为五个部分：预置模板、规则引擎配置、核心算法，底层求解能力。

1) 预置模板分行业及问题抽象问题模板，方便用户高效使用：

a) 行业模板：加工设备类场景及复杂人员调度类场景 ；

b) 标准模板：作业调度JSP、柔性作业调度FJSP、流水车间调度、单机调度、并行机调度、带有组批的作业车间调度、有缓存限制的作业车间调度及无等待时间的作业车间调度。

2) 规则引擎配置层，开放接口供用户配置或选择优化目标、生产约束及算法：

a) 文件配置：通过输入文件的形式配置；

b) SDK：提供C++接口供用户设置优化目标，生产约束及算法选型等。

3) 模型库，提供丰富的优化目标及生产约束支持：

a) 优化目标：完工时间、总加权流动时间、最大延迟时间、总加权拖期时间、加权拖期工件数、总机器负载、最大负载、能源消耗及碳足迹；

b) 生产约束：释放时间约束、滞后时间约束-最小时间间隔、滞后时间约束-最大时间间隔、无等待时间、固定不可用时间段、时间窗内不可用时间段、中断后可恢复、并行批处理、切换时间、有限缓冲区及任务阻塞。

4) 算法库，提供高效求解算法，具体包含：

a) 表调度、负载均衡、二阶段组批；

b) 局部/禁忌搜索、混合进化；

c) 数学规划。

5) 求解器，底层高效求解：

a) 数学规划求解器：混合整数规划、约束规划求解；

b) 启发式算法库：基础搜索算法；

c) 黑箱优化算法库：提供高效调参能力。

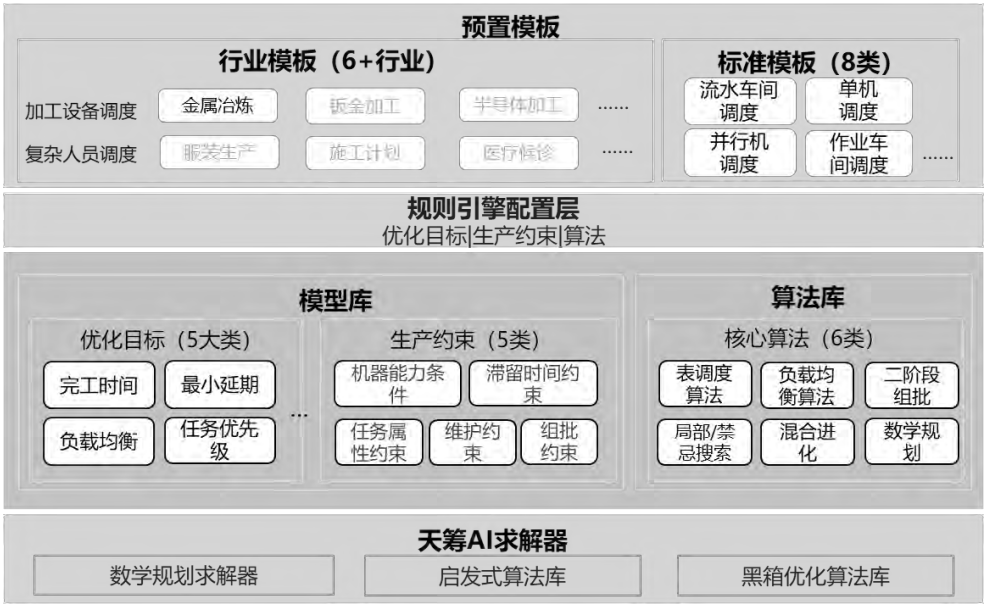

图3.6 生产调度求解器技术架构图

3.2.4 生产调度求解器求解流程

生产调度问题求解技术的关键流程可归纳为以下核心步骤：

1) 问题建模

a) 根据具体调度场景选择合适的问题模型；

b) 约束条件梳理；

c) 数学表示示例（以FJSP为例）：

i. 作业集合: $J = \{J_1, J_2, \dots, J_n\}$

ii. 机器集合：$M = \{M_1, M_2, \dots, M_m\}$

iii. 工序集合: O_{ij}表示作业J_i的第j道工序

iv. 加工时间：T_{ijk}表示工序O_{ij}在机器M_k上的加工时间

2) 目标函数定义

根据优化需求选择单一或多个目标，常见目标包括：

i. 最小化最大完工时间（MakeSpan）

$$\min C_{max} = \max\{C_1, C_2, \dots C_n\}$$

其中C_i 为作业 J_i 的完工时间。

ii. 最小化总延迟时间

$$\min \sum_{i=1}^{n} \max(0, C_i - d_i)$$

其中d_i为工件J_i的交货期。

iii. 多目标优化, 如需同时优化多个目标，可采用加权和法或帕累托最优方法，例如：

$$\min w_1 \times C_{max} + w_2 \times Tardiness$$

其中w_1,w_2 为权重系数。

3) 求解方案选型，生产调度问题的求解策略选择主要考虑问题复杂度及求解规模

a) 精确算法：适用于小规模问题，计算性能开销大，例如MILP、CP约束规划、动态规划等；

b) 启发式/元启发式：适用于规模较大或较为复杂的NP-hard问题，例如禁忌搜索（TS）、遗传算法（GA）、模拟退火（SA）、蚁群/粒子群及混合算法（VNS）等。

4) 初始解构造，初始解构造是生产调度问题求解的基础，高质量的初始解可以大幅度提升算法初始优度，提升搜索效率，通用的初始解构造方法主要有以下几类，具体选择取决于问题类型（如作业车间JSP、流水车间FSP、柔性车间FJSP）和优化目标：

a) 基于优先规则的构造方法，通过设定优先级快速生成可行解，常见规则包

括：

i. 最短加工时间优先：优先安排加工时间最短的工序。

ii. 最早交货期优先：优先处理交货期紧迫的工件。

iii. 先到先服务：按工件到达顺序分配。

iv. 临界比规则：动态调整优先级，适用于带交货期的场景。

b) 随机生成法，在满足工艺路线约束和机器占用约束的前提下，随机分配工序到机器或随机排列工序顺序。该方法简单但解质量可能较差 ；

c) 贪心算法，逐步构造解，每步选择当前最优操作：

i. 从待调度工序中，根据目标函数（如最小化空闲时间）选择最佳工序插入时间槽。

ii. 优先将工序安排到最早可用的机器上

d) 表调度，一种用于图表达和节点优先级的算法技术，用于产生兼顾多样性和优度的初始解。

5) 核心搜索策略

a) 精确算法：MILP建模后利用求解器求解；

b) 启发式/元启发式；

c) 禁忌搜索，通过禁忌表配合邻域结构跳出局部最优，通过优化禁忌表及邻域动作可答复提升搜索效率及优度；

d) 遗传算法，采用编码、选择、交叉和变异操作，常与局部搜索（如禁忌搜索）混合，形成GA+TS混合算法；

e) 变邻域搜索，动态切换邻域结构，结合扰动策略避免早熟收敛，自适应变邻域搜索可平衡全局与局部搜索能力。

6) 解评估

a) 快速评估，针对与图表达的部分问题中，增量评估失效的情况下，采用粗糙评估作为快速评估的实现方式，提升搜索效率；

b) 精确评估，相对耗时较长，但评估结果准确，间隔性进行，既保证搜索正确性，又兼顾搜索效率。

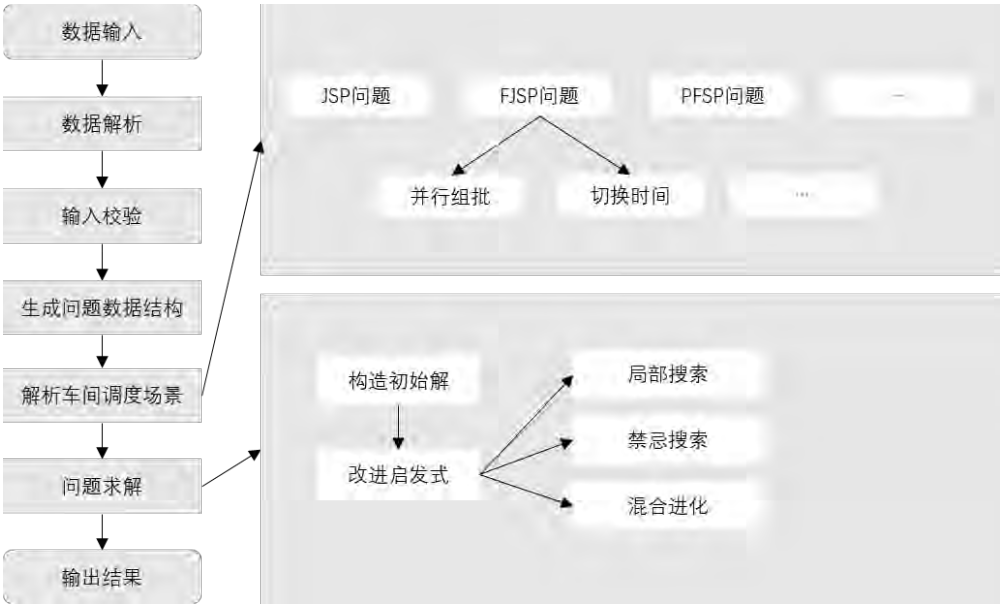

图3.7 生产调度求解器求解流程图

3.2.5 生产调度求解器功能特性

生产调度求解器作为为用户解决生产调度问题求解的工具，应提供基础的建模、参数设定、求解、结果获取等功能，也应提供回调、算法调参等高阶功能。

1) 模型支持：应支持典型的生产调度问题求解，作业调度、柔性作业调度、流水车间调度、单机调度、并行机调度、带有组批的作业车间调度、有缓存限制的作业车间调度、无等待时间的作业车间调度；

2) 优化目标支持：应支持设置优化目标为完工时间、总加权流动时间、最大延迟时间、总加权拖期时间、加权拖期工件数、总机器负载、最大负载、能源消耗及碳足迹；

3) 生产约束支持：应支持设置生产约束为释放时间约束、滞后时间约束-最小时间间隔、滞后时间约束-最大时间间隔、无等待时间、固定不可用时间段、时间窗内不可用时间段、中断后可恢复、并行批处理、切换时间、有限缓冲区及任务阻塞；

4) 求解能力：

a) 小规模问题精确求解，内置分支定界，MILP建模求解；

b) 大规模问题启发式求解，内置启发式及元启发式搜索算法。

5) 数据输入：应支持通过文件读入问题类型、优化目标及生产约束，应支持通

过SDK接口建模，应支持通过回调功能自定义产约束计算功能；

6) 日志和结果输出：应支持求解信息打印，包括输入模型简要信息，求解结果汇总（如优化目标值、求解时间、调度结果等；

7) 参数设置：应支持求解时间上限、算法选择、算法内参数；

8) 结果获取：应支持调度结果获取，调度结果甘特图显示。

3.2.6 生产调度求解器关键技术

3.2.6.1 析取图算法技术

析取图是一种用于建模和求解车间调度问题的强大图形化工具，它将复杂的调度约束转化为直观的网络图，尤其擅长表示作业车间调度问题。

1) 析取图的基本定义，一个标准的析取图模型通常表示为一个三元组：$G = (N, C, E)$。

a) N（节点集）：代表所有工序的集合。通常还包括两个虚拟节点：起始节点（Start/0）和终止节点（End/*）。每个节点（工序）的权重等于该工序在选定机器上的加工时间；

b) C（连接弧集）：是有向弧的集合，用实线表示。它描述了同一作业内部各工序之间由工艺路线决定的先后顺序约束；

c) E（析取弧集）：是无向弧的集合，用虚线表示。它连接了所有需要同一台机器加工的工序对。这些弧的方向是不确定的，调度求解的本质就是为这些析取弧确定方向（即决定在同一台机器上，哪个工序先加工），从而形成一个可行的调度方案。

2) 析取图的关键作用

a) 解的表示与可视化：析取图可以清晰地表示出一个完整的调度方案，将抽象的排序问题转化为直观的图结构；

b) 计算关键路径与最大完工时间：当一个调度方案确定后（即所有析取弧的方向被确定），析取图就变成了一个带权有向无环图。从起始节点到终止节点的最长路径即为关键路径，该路径的长度（各节点加工时间之和）就是整个调度方案的最大完工时间（Makespan）。优化调度的目标就是通过调整析取弧的方向来缩短关

键路径；

c) 邻域搜索的基础：在许多元启发式算法（如禁忌搜索、遗传算法）中，邻域操作通常基于析取图进行。常用的操作是在关键路径上反转一条析取弧的方向，从而生成新的邻域解，以期改进调度效果；

d) 合法性检验：一个可行的调度方案对应的析取图必须是一个有向无环图。如果图中存在回路，则说明存在逻辑冲突（如工序A必须在工序B之前，同时工序B又必须在工序A之前），是一个非法解。

3.2.6.2 混合算法求解技术

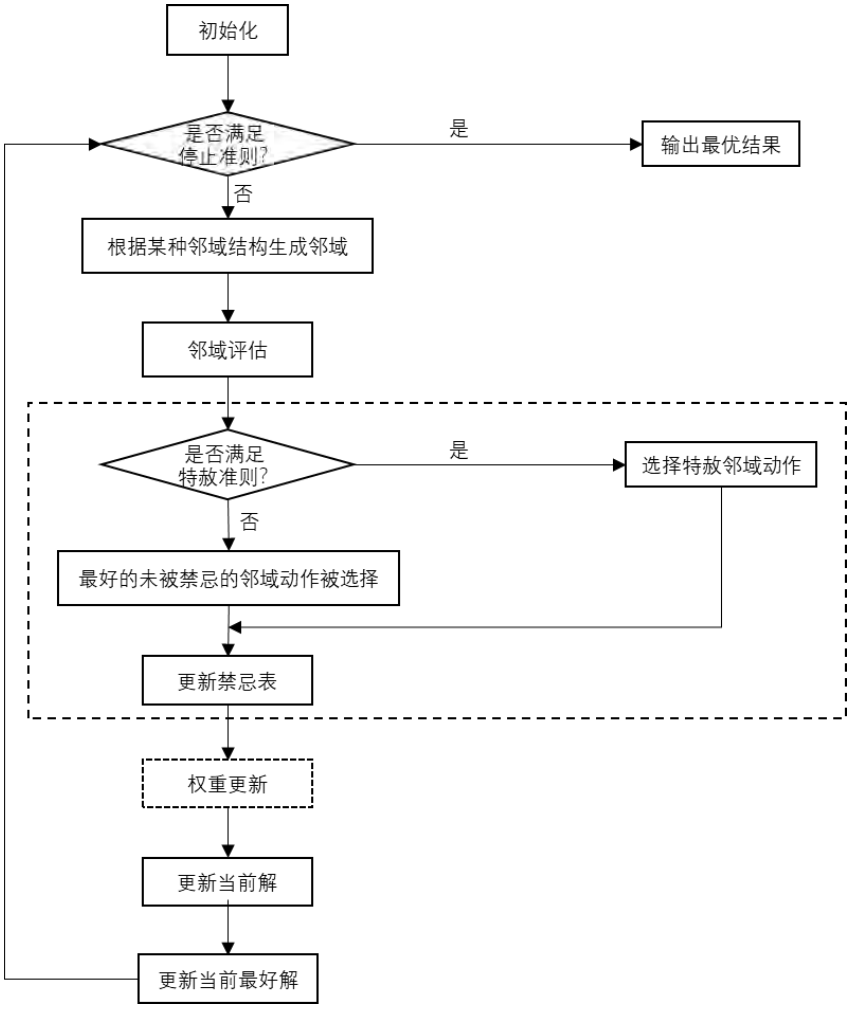

图3.8 生产调度禁忌搜索算法流程图

遗传算法与禁忌搜索混合算法GA+TS（Genetic Algorithm + Tabu Search）结合了遗传算法（GA）的全局搜索能力和禁忌搜索（TS）的局部优化能力。遗传算法

通过选择、交叉、变异等操作进行全局探索，但容易早熟收敛；禁忌搜索则通过禁忌表和特赦准则避免重复搜索，有效跳出局部最优。其基本流程为：遗传算法全局搜索 → 禁忌搜索局部优化 → 种群更新。典型实现步骤如下：

1) 初始化GA种群：生成多个排产方案作为初始个体；

2) 适应度评估：基于生产效率和稳定性构建目标函数；

3) 遗传操作：选择、交叉（双点交叉）、变异；

4) TS优化：将子代个体作为TS初始解进行邻域搜索；

5) 种群更新：用优化后的解替换适应度较差的个体。

3.2.6.3 MILP建模求解技术

问题描述：给定N个作业（Jobs）和M台机器（Machines）。每个工件$i(i = 1,2,\dots,N)$由一系列工序 $O_{i,1}, O_{i,2}, \dots O_{i,m(i)}$组成，其中$m(i)$是工件$i$的工序总数。每个工序$O_{ij}$必须在指定的机器$\mu(i,j)$上加工，加工时间为$p_{i,j}$。目标是安排所有工序在机器上的加工顺序，使得所有工件都完成加工的时间（即最大完工时间C_{max}）最小。

数学模型：

1) 决策变量 (Decision Variables)：

a) 连续变量：

$S_{i,j}$：表示工序$O_{i,j}$的开始加工时间；

$C_{i,j}$：表示工序$O_{i,j}$的完成加工时间；

C_{max}：表示所有工件的最大完工时间。

b) 0-1整数变量：

$X_{i,k,l}$：如果工序$O_{i,j}$在机器$\mu(i,j)$上的加工时间早于工序$O_{k,l}$（且 $\mu(i,j) = \mu(k,l)$，则$X_{i,k,l} = 1$；否则$X_{i,k,l} = 0$。

2) 目标函数 (Objective Function)：

最小化最大完工时间 C_{max}：$\min C_{max}$ 。

3) 约束条件 (Constraints)：

a) 工序完成时间计算约束：工序的完成时间等于：其开始时间加上加工时间。

$$C_{i,j} = S_{i,j} + p_{i,j}\ \forall i = 1,\ldots,N; \forall_j = 1,\ldots,m(i)$$

b) 同一工件工序顺序约束：每个工件的工序必须按照给定的工艺顺序进行。

$$S_{i,j+1} \geq C_{i,j}; \forall_i = 1,\ldots,N; \forall j = 1,\ldots,m(i) - 1$$

c) 同一机器资源约束（无重叠约束）：任意两个在同一台机器上加工的工序不能重叠。这通常通过引入0-1变量$X_{i,j,k,l}$和一个足够大的正数M（Big-M）来实现。设$\mu(i,j) = \mu(k,l) = m'$，即工序$O_{i,j}$和$O_{k,l}$都在机器m'上加工。

$$S_{i,j} \geq C_{k,l} - M \times \left(1 - X_{i,j,k,l}\right)$$

$$S_{k,l} \geq C_{i,j} - M \times X_{i,j,k,l}$$

这两个约束共同确保要么$S_{i,j} \geq C_{k,l}$（即$O_{k,l}$先于$O_{i,j}$），要么$S_{k,l} \geq C_{i,j}$（即$O_{i,j}$ 先于$O_{k,l}$）。M的选取应足够大，例如可以设为所有工序加工时间总和的一个上界。

d) 最大完工时间定义约束：最大完工时间C_{max}必须大于等于每个工件的最后一个工序的完成时间。

$$C_{max} \geq C_{i,m(i)} \forall_i = 1,\ldots,N$$

e) 时间非负约束：所有工序的开始时间非负。

$$S_{i,j} \geq 0 \forall_i = 1,\ldots,N; \forall_j = 1,\ldots,m(i)$$

f) 0-1变量定义约束：$X_{i,j,k,l} \in \{0,1\}$对于所有满足 $\mu(i,j) = \mu(k,l)$ 的工序对 $O_{i,j}, O_{k,l}$

这个模型通过定义连续变量（时间）和0-1整数变量（顺序），利用线性目标函数和线性约束条件（包括Big-M方法处理的工序互斥约束）来精确描述JSP问题。求解此MILP模型即可得到最优调度方案。对于更复杂的车间调度问题（如柔性作业车间调度FJSP、考虑维修的调度等），模型会在此基础上增加额外的变量和约束来描述其特有的复杂性，例如引入变量表示工序与机器的分配关系等。

3.3 运输计划求解技术

3.3.1 运输计划应用背景

运输计划是对车辆路径规划问题（Vehicle routing problem，VRP）落地提出行业求解方案，指的是通过科学的调度与路径规划技术，对运输资源（如车辆、人员等）进行合理配置，以确保货物从起点到达目的地的过程能够高效、经济、安全地完成[116]。运输计划需要根据需求、约束条件和环境因素，在最短时间内实现最佳的运输方案。

运输计划的核心目标是通过优化运输路径、合理分配运输任务、提高装载率等，降低运输成本，此外，也考虑降低车辆使用成本、降低阶梯计费载重成本和降低累计配送时间成本等目标。在上述目标的基础上，运输计划还能够同时处理复杂的约束条件，将车辆属性、时间要求、路网特点、仓库属性和订单特性纳入计划范畴，在确保运输计划结果可执行的情况下，在合理时间内找出高质量的运输方案。通过支持多种目标和约束，运输计划能够在城市配送、长途配送、快递物流等场景发挥作用，也能支持不同场景下的定制化需求。

传统精确算法，如数学规划和分支定界等，通过系统的数学建模和计算，能够精确求解问题的最优解，但是计算资源高，难以解决复杂、大规模问题。因此启发式算法常用于实际求解大规模实际问题，通过模拟自然界的生物、物理行为，对解的空间进行搜索，可以高效的给出高质量的解方案，但不保证解的最优性[117]。

3.3.2 运输计划问题分类

在经典的VRP落地实践中，按网络结构[118]（单层/多层/多中心）、资源约束[119]（载重/容积/车型异构）、交付限制[120]（硬时间窗、取送一体）等进行分类[121][122]。经典模型如载重受限VRP（Capacitated Vehicle Routing Problem，CVRP）与带时间窗VRP（Vehicle Routing Problem with Time Windows，VRPTW）构成算法逻辑的基础，进一步通过引入多配送中心、动态路网、装载约束等特性逐步对接实际业务的复杂性。此类分类体系为路径规划算子的设计与约束处理提供了标准化框架，但需结合城市配送、干线物流或冷链运输等具体运输场景的行业规则进行参数化适

配[123][124]。具体问题分类详见表3.4。

表3.4 经典运输计划问题分类

问题名称	英文全称	问题描述
CVRP	Capacitated Vehicle Routing Problem	车辆具有载重限制的 VRP
VRPTW	Vehicle Routing Problem with Time Windows	订单具有指定交付时段的 VRP
MDVRP	Multi-Depot Vehicle Routing Problem	车辆具有多始发配送中心的 VRP
VRPB	Vehicle Routing Problem with Backhauls	订单需要先送货后取货的 VRP
VRPPD	Vehicle Routing Problem with Pickup and Delivery	客户区分为取送货的 VRP
VRPSPD	Vehicle Routing Problem with Simultaneous Pickup and Delivery	客户同时具有取送货需求的 VRP
3L-VRP	Three-Dimensional Loading Vehicle Routing Problem	车辆车厢考虑货物立体三维装载约束的 VRP
HFVRP	Heterogeneous Fleet Vehicle Routing Problem	车辆具有不同载重等属性的的 VRP
OVRP	Open Vehicle Routing Problem	车辆完成服务不必返物流中心的 VRP
MTVRP	Multi-Trip Vehicle Routing Problem	车辆需要执行多轮次配送的 VRP
2E-VRP	Two-Echelon Vehicle Routing Problem	具备中央仓库、装运仓库两级配送网络的 VRP
TDVRP	Time-Dependent Vehicle Routing Problem	车辆行驶速度等与时间相关的 VRP
VRPSD	Vehicle Routing Problem with Stochastic Demand	客户具有随机需求的 VRP
OPVRP	Orienteering Problem Vehicle Routing Problem	访问部分客户最大化收益的 VRP
GVRP	Green Vehicle Routing Problem	考虑燃油排放等的绿色 VRP

3.3.3 运输计划求解器软件架构

运输计划求解器采用分层架构，将业务建模、算法求解与工程运行解耦，确保系统具备良好的可维护性、可扩展性和算法演进能力。如图3.9中的求解器软件架构所示，运输计划求解器主要划分为四个核心层次：业务与模型抽象层、求解调度与控制层、核心算法引擎层和通用能力支撑层。

业务与模型抽象层的核心职责是将复杂、多样化的运输业务需求，转化为标准化、可计算的运输计划问题模型。它负责业务数据接入，包括公有云、数据IO和SDK形式，并负责抽象业务对象、各类约束条件和定义统一的成本函数体系。它将算法和业务来源隔离，保证业务规则按照统一的标准模型进入算法。

求解调度与控制层是运输计划求解器的中枢，它负责算法策略选择，即在不同规模、不同约束组合下，选择合适的算法或算法组合；求解流程编排，将初始解生成、全局搜索、局部精修、解收敛判断组合；终止与稳定性控制，限制算法的运行时间与解的收敛判断。

核心算法引擎层式求解器的核心计算能力所在，提供多种成本模型，包括车辆成本、阶梯计费、累计用时以及服务收益等，具备多种约束模型，覆盖技能约束、时间约束、可达约束等，提供主流的启发式和精确求解方法，并提供统一的可行性校验、解的表示等周边能力。

通用能力与运行支撑层为整个求解器提供工程级支撑能力，一方面提供数学规划求解器和启发式算法库等能力，另一方面，提供高效的数据结构与内存管理、并行计算与多线程支持、日志、监控、性能统计、参数配置和实验复现能力，保证求解器在真实场景下的稳定运行，以及算法性能的可评估、可调优和持续演进。

图3.9 运输计划求解器软件架构图

3.3.4 运输计划求解器求解流程

运输计划求解器旨在实现从结构化数据输入到高质量调度方案的全链路转化，主要的求解流程见图3.10。该过程基于引擎内置的建模机制，针对特定业务变体构建精确的数学模型，并依托高效求解技术完成路径优化与结果映射。

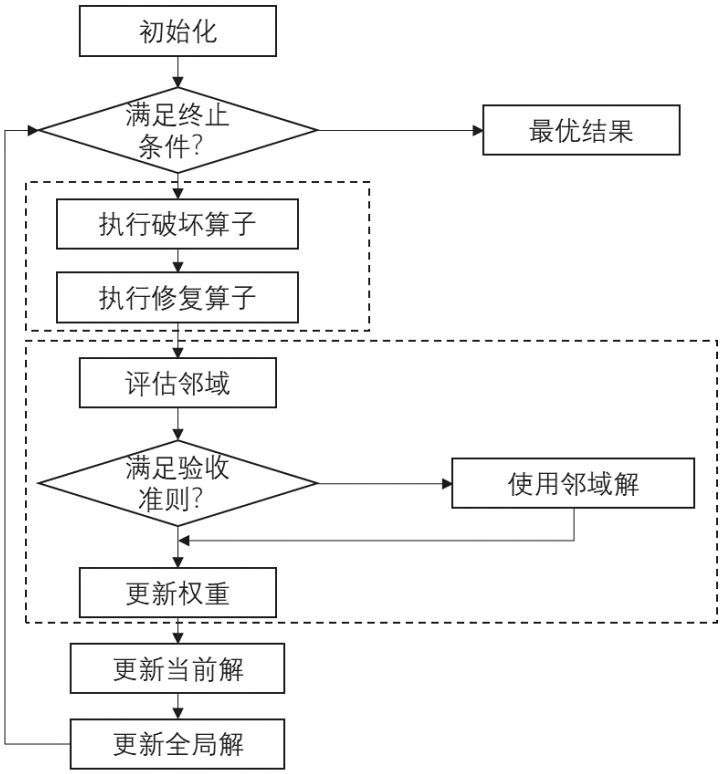

图3.10 运输计划经典求解流程示意图

具体执行阶段如下：

1) 问题感知：问题数据输入到运输计划引擎时，首先对数据特征进行识别。

a) 规模识别：通过订单数量判断数据规模，以选择精确算法或者启发式算法。

b) 约束拓扑识别：识别硬约束组合。例如：是否带时间窗（VRPTW）、是否多仓（MDVRP）、是否有取送一体（VRPPD）等。

c) 时效性识别：问题类型是“即时配送”还是“次日计划”，从而判断算法允许的执行用时。

2) 场景与算子匹配：依据识别到的问题特征，引擎从算法仓库中调配资源。

a) 模型路由：基础配送直接使用标准的 CVRP 模型；如果具备交付时间段要求，则匹配VRPTW模型等。

b) 算子初选：针对约束特点，匹配针对性算子，如订单分布密集的数据，使用聚类破坏算子等。

3) 核心求解阶段：该阶段执行对问题解空间的探索。

a) 求解层：求解核心由初始化构造与约束修复逻辑组成。该阶段综合运用贪婪启发式算法、模型松弛及约束处理技术，能够高效的构建满足载重、时间窗等硬性约束的初始可行解。

b) 优化层：采用局部搜索和迭代策略一步优化，在全局范围内通过跨线路交换、重插入等算子持续降低总运输成本，寻求求解质量与计算耗时之间的最优平衡点。

4) 方案后处理：在达到预设的收敛准则或时间上限后，求解器将最优路径方案、车辆利用率及成本分解等数据封装为标准 JSON 格式输出。

a) 可行性诊断：计算最终方案，对于无解情况，给出“冲突点”，告知所给数据异常，以免给出异常求解方案。

b) 业务指标对齐：将问题目标之外的重要业务参数进行计算。

3.3.5 运输计划求解器功能特性

运输计划求解器作为领域落地的解决方案，为实际客户解决痛点问题，提供建模、参数设定、求解、结果展示等多种能力，也支持算法调参等能力。

1) 模型支持

支持多种典型VRP问题求解，对应的运输计划模型有带容量运输、客户分组运输、订单拆分运输和单车运输等。

2) 目标支持

支持多种优化目标，包括距离成本、车辆使用成本、累计配送时间、载重成本和阶梯计费成本。

3) 约束支持

支持多约束条件下的优化求解，包括车辆载重约束、容积约束、最大作业约束、补能约束；客户时间窗约束、耗时约束、时间依赖约束；仓库车辆数量约束、可服务约束；订单取送货约束、需求分组约束、拆分配送约束和收益最大化约束。

4) 求解功能

支持多种求解方式，针对小规模时间不敏感场景，使用精确算法求解，针对大规模时间敏感场景，则使用启发式算法进行求解。

5) 数据输入

支持多种文件读入形式，支持输入订单数据、车辆数据、路网数据和业务配置数据，支持使用结构化文件设定问题数据、求解目标等，支持使用SDK设定求解器参数，例如求解时间、内置启发式算法类型。

6) 日志与结果输出

支持求解信息展示，包括求解过程日志、性能统计信息、结果结构化输出等。

7) 参数配置与调优机制

支持调整算法层参数、约束惩罚参数和求解策略参数等。

3.3.6 运输计划求解器关键技术

3.3.6.1 启发式算法求解技术

启发式算法是一种用于求解组合优化问题的经典方法，特别适用于处理大规模、复杂的优化问题。它通过在搜索空间中适应性地扩展和缩小邻域范围来改进解的质量，进而找到近似最优解。启发式算法结合了多种邻域搜索策略，并且通过调整邻域的选择概率来增强搜索的多样性和效率。启发式算法的核心在于“破坏-重建”构造邻域，并通过搜索结果自适应选择邻域，提高搜索效率。其经典步骤如下：

1) 初始化：生成一个初始解，并为每个邻域操作分配一个初始选择概率；

2) 迭代过程：

a) 在每次迭代中，选择一个邻域操作进行解的破坏；

b) 破坏后，使用某种启发式方法对解进行重建，生成一个新的解；

c) 评估新解的质量，并根据质量来决定是否接受新解。如果新解较好，则接受新解并更新当前解;如果新解较差，则可能以某种概率接受该解，以避免算法陷入局部最优。

3) 更新邻域操作的选择概率：

根据每种邻域操作的表现（如是否改善解的质量），调整该操作的选择概率，表现较好的操作会有更高的选择概率。

4) 终止条件：

根据设定的终止条件（如最大迭代次数、计算时间限制、或者解的质量停止改善）来终止算法。

3.3.6.2 MILP求解技术

混合整数线性规划（Mixed-integer linear programming，MILP）涉及对问题的数学建模，本节以带容量约束的车辆路径问题（Vehicle Routing Problem with Capacity，CVRP）为例，对MILP求解流程进行阐述。本节的容量与车辆最大载重含义相同。

1) 问题描述

给定一组客户和一个中央仓库，每个客户有一定的需求，每辆车有一个固定的容量，为车辆分配若干待服务客户，车辆装载对应客户的货物，从中央仓库出发，通过一系列的路线服务所有客户后返回，使得每个客户被服务一次，并且车辆的总行驶距离最小。

这里，假定所有车辆车型（包括但不限于容量、行驶速度、装卸货时间和服务水平）一致，且车辆数量充足，能够服务所有客户。

2) 集合与参数

N ：节点集合（即仓库、客户节点，默认$n = 1$表示仓库，$n = 2, \dots, N$为客户）

K ：车辆集合

$i \in 1, \dots, N$ ：节点索引，涉及两个节点时，使用j用于区分

$k \in 1, \dots, K$ ：车辆索引

q_i ：表示客户i的需求量

$d_{i,j}$ ：表示客户i和客户j之间的距离

Q ：表示每个车辆容量

3) 决策变量

$x_{i,j,k} \in \{0,1\}$ ：表示车辆k是否从节点i行驶到节点j

$u_{i,k} \geq 0$ ：车辆k离开节点i时的累计送货量

4) 目标函数

最小化所有车辆的总行驶距离，即：

$$Min\sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N}d_{i,j}x_{i,j,k}$$

5) 约束条件

流平衡约束条件，即每个车辆进入节点j后最终离开节点j。换言之，所有车辆在服务完客户后，必须离开该客户，所有车辆从仓库出发之后，必须返回仓库：

$$\sum_{i=1}^{N}x_{i,j,k}=\sum_{i=1}^{N}x_{j,i,k}$$

客户访问唯一性约束，即除仓库外的所有节点仅进入一次。换言之，所有客户的需求，必须是一辆车通过一次服务完成：

$$\sum_{k=1}^{K}\sum_{i=1}^{N}x_{i,j,k}=1,j\geq 2$$

仓库出发约束，即所有车辆均从仓库出发。换言之，所有车辆必须从仓库出发，且每个车辆只会执行一组配送任务，不会多次返回仓库重新出发执行配送：

$$\sum_{j=2}^{N}x_{1,j,k}=1$$

车辆容量约束，即车辆服务的客户的总需求不能超过车辆的容量。换言之，每个车辆分配到的客户的货物总重量，不能超过车辆的载重上限，因为车辆在仓库时会将所有客户的货物装车，默认拒绝超载：

$$\sum_{i=1}^{N}\sum_{j=2}^{N}q_j\,x_{i,j,k}\leq Q$$

载重连续性约束，即车辆服务客户后的累计交付载重应大于等于服务上一个客户。换言之，该约束保障求解得到的路线必须是从仓库出发，经过一组不重复的客户之后，返回仓库的正常路线，防止路线结束时回到已经服务过的客户：

$$u_{j,k}-u_{i,k}\geq q_j-Q\left(1-x_{i,j,k}\right),\forall i,j\neq 1,i\neq j$$

$$0\leq u_{i,k}\leq Q$$

如图3.11所示，异常路线满足除了载重连续性约束以外的所有约束，通过连续性约束排除这个可能性。

(a) 正常路线 (b) 异常路线

图3.11 违背连续性约束路线示意图

⚠黑色三角表示仓库节点，①黑色圆点表示客户节点，→ 实线表示正常路线，⇢ 虚线表示异常路线

该模型以最小化车辆总行驶距离为目标，限制每个客户必须被服务一次、每辆车从仓库出发并返回、每辆车的载重不能超过其容量，并通过连续性约束（Miller-Tucker-Zemlin，MTZ）消除子环路，从而准确将问题转换为数学模型，通过求解这个MILP模型就可以得到最优的运输计划。对于复杂的问题场景，模型通过增加集合、参数、决策变量，以及增改约束条件和目标函数完成描述。

3.4 二维切割求解技术

3.4.1 二维切割应用背景

二维切割问题（又称装箱问题，两者在数学本质上一致）旨在将一系列二维矩形或不规则零件互不重叠地放置在原材料上，以最大化原材料利用率。该算法在工业生产中应用广泛，其中家具制造和服装生产是两个典型领域。家具行业主要采用电子锯进行切割，加工零件多为矩形，原料为固定尺寸的矩形板材。其核心需求是适配“叠版”和“一刀切”工艺模式，在满足生产约束的同时，兼顾板材利用率与加工效率的提升。服装行业是二维切割算法的重要应用场景，其排版约束更为复杂，主要体现在三个方面：一是零件约束，服装零件多为不规则形状，轮廓复杂、差异显著，对算法的几何处理能力提出较高要求；二是原料约束，布料通常宽度固定、长度可视为无限；三是工艺约束，需严格匹配面料纹理方向，避免因排版方向错误影响服装品质。

二维切割问题属于经典的NP-Hard问题，其求解难度随问题规模扩大呈指数级增长。由于不存在多项式时间内的精确求解算法，当处理大规模实例时，精确方法

如枚举或分支定界往往因计算资源消耗过大而无法满足实际生产的时间要求。不规则零件排样问题尤为复杂，其边界轮廓的多样性导致几何运算（如旋转和碰撞检测）复杂度显著高于矩形排样，成为学术界和工业界的公认难题。因此，针对大规模实例的求解，业界普遍采用启发式或智能搜索算法（如遗传算法、模拟退火算法等），以在原材料利用率与求解效率之间取得平衡，既满足行业精度要求，又适应生产实时性需求。

3.4.2 二维切割问题分类

切割问题分类有四大核心维度：一是空间维度，分为一维、二维和三维；二是分配类型，分为强分配和弱分配两类，强分配要求所有给定的零件必须无遗漏地全部排在原料中，弱分配仅需从零件集合中选择最优零件子集完成分配，无需全部装入；三是原料种类，按照种类异质性分为：单一规格，即所有原料的规格相同；弱异质：原料种类数量有限，每个种类原料数量有限；强异质：原料有多种种类，每种种类数量无限；四是零件种类，按照种类异质性分为：弱异质：零件种类数量有限，每个种类有多个相同零件；强异质：几乎每个零件尺寸和属性都不相同[125]。

其次，基于优化目标，切割和装箱问题可分为输入最小化与输出最大化两大核心类别。输入最小化问题核心目标是最小化用于装下全部零件所需的原料，输出最大化问题核心目标是在给定的有限原料约束下，最大化装入的零件总数量或总价值等。

二维切割问题是工业场景中应用最广泛的切割和装箱问题，绝大多数场景属于强分配和输入最小化问题，即需将所有给定的二维小零件无重叠排布于原料中，最小化原料总使用量。基于原料种类维度，二维切割问题可分为两大核心子类：一是常规二维下料问题，即原料的尺寸固定，优化目标为最小化原料使用数量；二是开放维度问题，即原料的宽度为固定值，原料的长度可认为是无限长，优化目标为最小化原料的总使用长度，同时保证所有零件无重叠装入原料中。开放维度问题广泛应用于服装和金属卷材的切割场景[126]，是二维输入最小化问题中研究最活跃的分支之一，本文将详细介绍二维开放维度问题的求解流程和关键技术。

3.4.3 二维切割求解器技术架构

二维切割的技术架构如图3.12所示，主要包括四个层次：预置行业模板、引擎配置层、模型库与算法库、天筹AI求解器。此四层架构既提供了强大的底层求解器，保证求解算法性能，同时通过预置行业模板和特性配置层，解决不同行业和用户场景适配难题，实现开箱即用的交付体验。

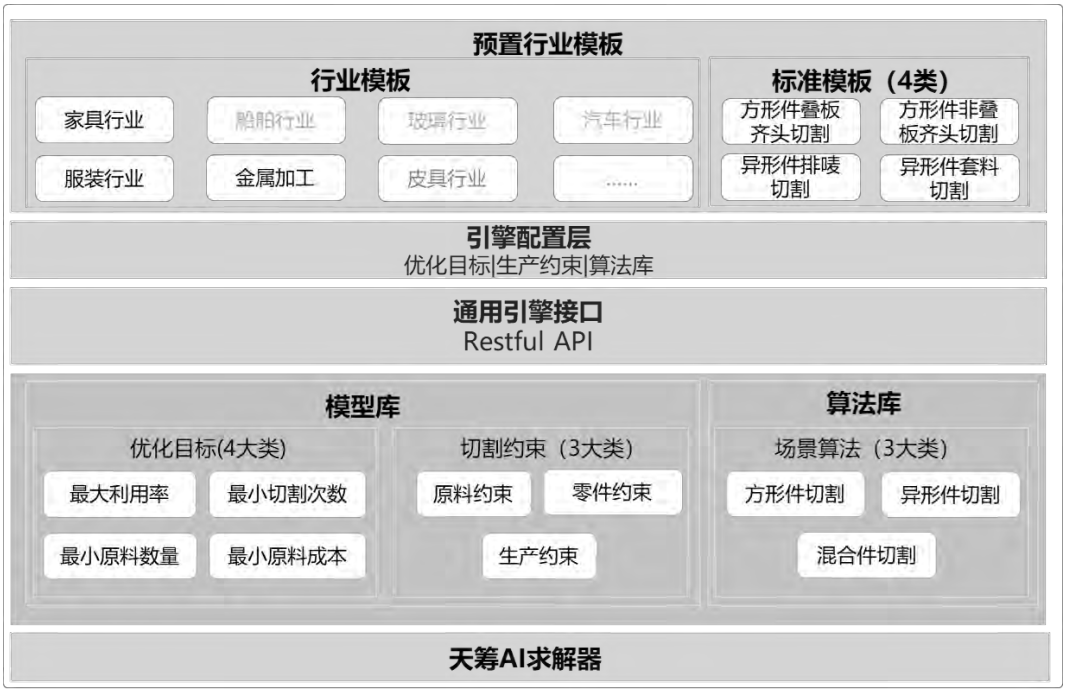

图3.12 二维切割求解器技术架构图

1) 预置行业模板

系统预设了多个行业模板，如家具行业、服装行业和金属加工行业等，每个模板针对特定行业特点进行构建。如家具行业的“叠版”和“一刀切”等生产约束，服装行业的“分行分列”和“同套同方向”约束。这些模板为不同行业提供了行业场景级的解决方案。

2) 引擎配置层

引擎配置层可以根据行业生产特点配置切割算法的优化目标、生产约束和算法库，确保引擎能够根据行业场景实际生产要求进行灵活配置。

3) 模型库与算法库

模型库包含优化目标和切割约束的详细分类，优化目标包括最大化原料利用率和最小化加工时间等；切割约束包括：原料约束：原料形状和个数等，零件约束：

零件形状、旋转角度和翻转等约束，生产约束：行业特定的生产约束。算法库则包括针对零件是方形件、异形件和混合件开发的专业求解算法。

4) 天筹AI求解器

作为底层核心，包含数学规划求解器、启发式算法库和黑箱优化算法库，为系统提供强大的计算能力和优化能力，确保二维切割的高效性和准确性。

3.4.4 二维切割求解器求解流程

本节针对二维服装切割引擎求解流程进行详细说明，即零件是不规则形状，原料定宽无限长场景。问题描述如下：给定原料和零件，将零件不重叠地排布在原料里，使得原料的利用率最高。因为原料的宽度事前给定，因此将优化目标转化为最小化原料长度。二维异形件切割问题数学建模如下[127]：

$$
\begin{aligned}
&minimize && L \\
&subject\ to && (P_i\ (o_i) \oplus x_i) \cap \left(P_j\ \left(o_j\right) \oplus x_j\right) = \emptyset, 1 \le i \le j \le n \\
& && (P_i\ (o_i) \oplus x_i) \subseteq C(W, L), 1 \le i \le n \\
& && L \in R_+ \\
& && o_i \in O_i, 1 \le i \le n \\
& && x_i \in R^2, 1 \le i \le n
\end{aligned}
$$

其中，L表示使用原料长度，W表示原料宽度，P_i表示第i个零件，o_i表示第i个零件的旋转方向，O_i表示第i个零件的可能旋转方向集合，x_i表示第i个零件参考点放置坐标。

上述问题包含3个决策变量，分别是L, x和o。算法选择固定其中的两项L和o，在此基础上优化x。在上面的问题模型中，要求零件之间不发生重叠，以及零件要放置在原料内。当原料长度为L时，若得到了可行解，则缩小原料长度，若没有得到可行解，则适当增加原料长度。之后，扰动当前解，实施上述优化程序。该过程迭代进行，直到达到算法终止条件。最终选择原料长度最小的可行解作为算法的输出。算法的求解流程如图3.13所示：

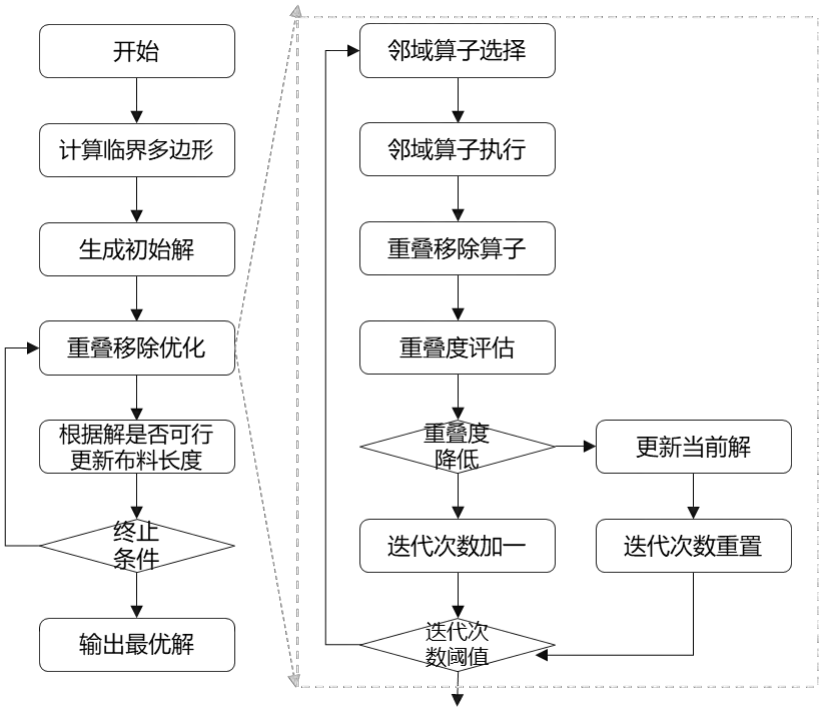

图3.13 二维切割求解器求解流程图

其中，核心求解技术要点在于重叠移除算法，包括邻域算子和重叠移除算子。

1) 邻域算子

本算法采用“2交换”作为邻域搜索算子[128]，也即随机交换2个零件的位置，其操作流程如下：

a) 随机生成需交换的零件对i, j；

b) 将零件i移出当前排样方案；

c) 保持零件j不动，复制一个零件j，尝试不同旋转或翻转状态的零件j,将其放置在与其他零件重叠面积最小的位置，然后移除原零件j；

d) 尝试不同旋转或翻转状态的零件i,将其放置在与其他零件重叠面积最小的位置。

2) 重叠移除算子

本引擎使用嵌入深度度量两个零件的重叠量。两个重叠零件之间的嵌入深度定义为使两个零件消除重叠所需的最小位移。可将零件P_j和零件P_i分离的最小移动方式为将零件P_j沿水平方向或者竖直方向移动，直至零件P_i和P_j恰好不发生重叠，该位移向量即为最小位移向量，向量长度即为两个零件之间的嵌入深度[127][129]。

3.4.5 二维切割求解器功能特性

二维切割求解技术支持如下11个主要特性约束（见表3.5），分类如下：

1) 原料约束

a) 原料形状：原料宽度固定，要求排版零件的总宽度不得超过原料宽度；

b) 避瑕疵约束：排版时需避开原料上的破洞、污渍等疵点，避免疵点切入零件导致成品不合格；

c) 锁定排料：排版时将某零件固定在原料位置上，以便保证原料图案固定在特定零件上，多见于服装面料裁剪。

2) 零件约束

a) 零件形状：支持任意简单多边形的异形零件形状；

b) 零件旋转：由于原料由经纬线组成，支持零件可以旋转90度和180度两种角度；

c) 零件翻转：由于原料表里区别，支持零件可以沿x、y轴翻转；

d) 组合排料：两个或多个零件预先组合在一起，排料结果中相对位置保持不变；

e) 同套同方向排料：相同尺码的零件的旋转和翻转方向保持一致。

3) 生产约束

a) 零件间距：零件之间预留间隙余量，避免切割时伤到零件；

b) 分行排料：不同套零件放置原料上下不同区域；

c) 分列排料：不同套零件在原料的左右不同区域。

表3.5 二维切割求解器特性列表

序号	特性约束	详细说明
1	原料形状	支持矩形原料
2	零件形状	支持矩形以及异形零件
3	零件旋转	零件90、180度旋转
4	零件翻转	零件翻转
5	零件间距	零件之间间隔相等距离
6	锁定排料	零件固定在原料上
7	组合排料	控制多个零件之间的位置关系
8	避瑕疵排料	自动避开原料上设定的瑕疵区域
9	分行排料	不同套零件在原料上下不同区排版

10	分列排料	不同套零件在原料左右不同区排版
11	同套同方向排料	同套零件有相同的旋转角度

3.4.6 二维切割求解器关键技术

二维异形件排样算法涉及的关键技术主要有两个：高效计算几何算法和排样策略。

3.4.6.1 高效计算几何算法

计算几何问题是二维异形件排样的一个核心挑战，其主要内容在于计算不规则形状的靠接位置、确定零件与原料之间的包含关系、判断是否重叠以及实现二维区域之间的交、并、差等布尔运算。不规则形状的表示一般有两种方法，一种是位图点阵表示法，以像素的形式来表示图形。该方法算法简单，但是属于非精确表示，其精度较低，而高精度下的计算时间无法满足实际使用要求，目前排样算法已基本不采用此方法来表示图形。另外一种采用较多的形状表示方法是几何图形表示法，其中应用广泛的是使用临界多边形的概念（No-Fit Polygon, NFP），其简要定义如下：给定两个多边形，其中一个固定，另一个多边形围绕固定的多边形作平移的刚体运动，并围绕固定多边形滑动，直到回到起点位置，在此过程中在运动多边形上选取一点作为参考点，则参考点在环形运动过程中形成的轨迹就称为临界多边形。如图3.14所示，若多边形A放置位置固定，则多边形B相对于A进行滑动形成的NFP记为NFP_{AB}[130]。

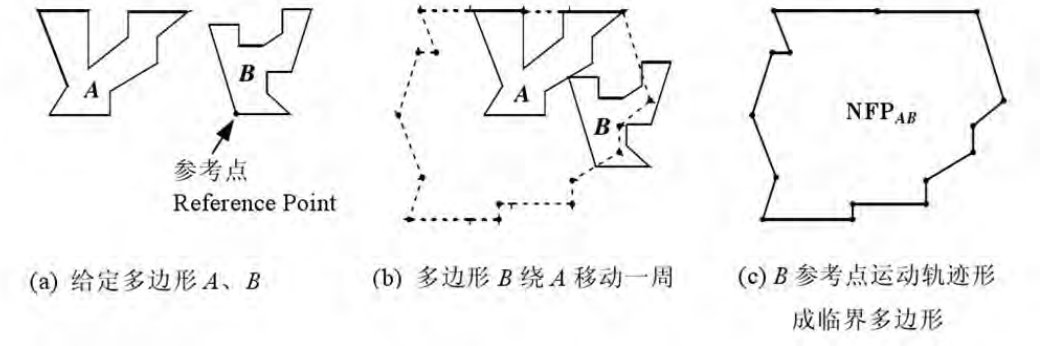

图3.14 临界多并行（NFP）概念示例说明

3.4.6.2 排样策略

排样策略是决定排样效果的另一个至关重要的方面。目前学术界及业界采用的排样策略大体上可分为三类，包括：基于混合整数规划的排样策略，基于可行解的构造式排样策略和基于重叠移除技术的改进式排样策略。

基于混合整数规划的排样策略通过将排料问题建模为混合整数规划模型，并借助数学规划求解器进行求解。该策略的核心难点在于不规则图形的数学表达，目前主流的建模方法包括：基于像素点的离散化建模、基于圆盘覆盖的裁片逼近建模，以及基于D函数的多边形精确表达建模。这些方法在不同精度与计算复杂度之间提供了多种建模选择。虽然基于混合整数规划的排样策略理论上可以求得最优排样方案，然而由于这类模型自身复杂度过高，在合理时间内仅能解决有限规模的问题，因此实用性不强[131]。

基于可行解的排样策略是一种构造性算法，其核心流程是从空白原材料出发，按照预设顺序逐个放置零件，直至完成全部零件的布局。在排样过程中，零件仅被放置在满足约束条件的位置，且一旦放置，其位置与旋转角度即被固定，从而始终保持整个排样方案处于可行状态。该策略的排样效果主要受两个关键因素影响：一是零件的放置次序，二是每个零件具体的放置规则（包括旋转角度和摆放位置的选择）。这两个因素也成为现有基于可行解的排样算法的主要优化方向。在零件排序方面，通常采用“先大后小、先难后易”的原则，优先放置面积较大、形状复杂或难以排布的零件，随后利用形状规则、面积较小的零件填充剩余空隙。常用的排序依据包括零件面积降序、零件宽度降序，以及零件外包矩形面积降序等。在放置规则方面，算法需根据评价标准为每个零件选择最优的旋转角度和摆放位置。评价标准通常分为两类：位置评价标准（如“左底”策略、贴近边缘原则、最小化使用长度等）和契合度评价标准（如零件外包矩形重叠面积、外拓图形重叠面积、几何中心距离等）。实际应用中往往需综合考量这两类标准以提升材料利用率。由于其良好的实用性与可扩展性，基于可行解的排样策略已成为当前应用最为广泛的排样方法之一。其优点在于求解速度快，可在较短时间内得出较好的排样结果[132]。

基于重叠移除的排样策略是一类允许零件在排样过程中发生重叠，随后利用分离技术逐步消除重叠、从而提升原材料利用率的优化方法。衡量两个零件之间重

叠程度的主要指标包括重叠面积、最小嵌入深度，以及最小横向/纵向嵌入深度等。通过合理选择重叠评价指标和分离策略，该策略可有效改善排样布局的紧凑性。重叠移除算法的基本思路是，给定一个可行排样方案后，缩短原料长度，此时会有部分零件超过原料边界，将超过原料边界的零件向左移动，此时该零件将和部分零件发生重叠，采用重叠移除算法重新排列零件，使得零件之间重叠指标最小化，重复上述步骤，进一步减少零件重叠指标，直到完全消除重叠为止[127][129]。

3.5 三维装箱求解技术

3.5.1 三维装箱应用背景

三维装箱问题在仓储物流、运输配送及制造业等领域具有广泛的应用价值，其核心目标是通过算法优化实现空间利用率最大化与成本最小化。例如，在制造业仓储与运输场景中，通常需处理多维度约束，包括物品体积、重量、装卸顺序、易碎品隔离及重心平衡等，以保障运输方案的可执行性与安全性；在电商物流场景中，面对多规格、多形状商品的异构性，则需快速生成装箱方案以满足高频次配送的时效性需求。

三维装箱问题是仓储物流、运输配送及制造业中的核心优化难题之一。该问题可形式化描述为：给定n个物品（每个物品具有长、宽、高等三维属性及可能的重量约束）和m个容器（载具），目标是在满足容器容积限制、物品不重叠、载具承重限制等约束条件下，实现容器数量最小化与空间利用率最大化。该问题是组合优化领域的经典NP-Hard问题，即在多项式时间内无法保证找到全局最优解。其计算复杂度随物品数量增加呈指数级增长，主要挑战包括：

1) 组合爆炸：物品的摆放顺序、旋转方向（6种空间朝向）及摆放位置的选择构成巨大的搜索空间；

2) 多约束耦合：需同时考虑几何约束（如物品不重叠）、物理约束（如重心平衡）及业务约束（如装卸顺序）等。

3.5.2 三维装箱问题分类

三维装箱问题可根据研究维度、约束条件及优化目标等，分成不同类别。系统化的分类有助于研究人员和从业人员更清晰地界定问题边界，选择合适的求解方法。

在学界中，早期的Dyckhoff分类[133]将问题按照问题维度、单容器/多容器、物品异构性等进行分类，以切割与包装问题的几何特性为基础，建立了早期的分类框架。该分类方式后续被Wäscher等人[125]改进优化：1) 第一层先基于优化目标（输出最大化、输入最小化）进行划分； 2）第二层再基于箱子的异构性（单物品、同构、异构等）进行更细致的分类（如IIPP、SLOPP、MILOPP等），形成了更通用的新分类体系。

基础三维装箱问题通常仅考虑标准物理约束（如物品不可重叠、需适配容器空间等）。然而，实际工业场景（如集装箱装载、货车装载等）中的约束更为复杂多元。较为常见的实际约束如表3.6所示[134][135]：

表3.6 三维装箱求解器约束类别

约束类别	详细约束项	实际意义
容器相关	• 载重约束 • 重量分布约束	确保不超载且重量均衡分布
物品相关	• 装载优先级 • 可放置方向 • 可堆叠性	考虑物品特性和操作要求
订单相关	• 完整货物集 • 货物分配约束	保证订单完整性和分配合理性
位置相关	• 危险品隔离 • 相对位置要求 • 多卸货点顺序	满足安全规范和物流效率
装载相关	• 装载稳定性 • 方案复杂度	确保运输安全和操作可行性

随着约束条件的增加，三维装箱问题的计算复杂度显著提升：

1) 基础问题：NP难问题，多项式时间内无法得到全局最优解；

2) 多约束问题：需要设计专门的混合整数规划模型或元启发式算法；

3) 实时应用：在线装箱算法需要平衡求解质量和计算效率。

本文将重点介绍离线场景下兼顾多种真实约束的三维装箱问题求解流程和关

键技术。

3.5.3 三维装箱求解器技术架构

三维装箱算法架构可分为三个层次：预制输入模板、算法配置及辅助工具部分以及核心算法部分。如图3.15的技术架构所示，此三层架构既提供了多样的求解策略，保证求解算法性能，同时通过预置行业模板和算法优化目标和装载约束配置层，解决不同行业和用户场景适配难题，达到开箱即用的目的。

图3.15 三维装箱求解器技术架构图

1) 预置输入模板

预置模板提供不同行业场景模板和标准输入。场景模板包括集装箱装载、货车装载、航空箱装载、散件加托、小件装箱等，适用于不同工业场景。标准输入则涵盖了载具类型、箱子信息、托盘信息、箱子类型、路线信息等，制定统一的输入标准，可以适配不同场景的装箱需求。

2) 算法配置及辅助工具

算法配置层定义了优化目标、装载约束，并提供了辅助工具以增强算法的实用性和可靠性。优化目标包括使用最少的容器数、最小化成本和最大化装载率，满足多目标优化需求。装载约束分为物理约束、业务约束和路线约束，全面覆盖超过20项约束条件，确保装载方案的可行性和高效性。额外配置包括时间限制和超参数配

置，使算法能够灵活应对不同场景及不同的性能需求。辅助工具如输入检验、输出校验和后处理优化，进一步保障算法的准确性和鲁棒性。

3) 核心求解策略

核心求解策略层包含贪婪向前搜索策略、双层贪婪向前搜索策略和极点随机贪婪搜索策略。这些策略能够系统性地平衡空间、重量、堆叠等复杂物理与业务约束，快速生成高质量的装载方案，有效替代耗时的人工模拟，提升效率和决策科学性。

3.5.4 三维装箱求解器求解流程

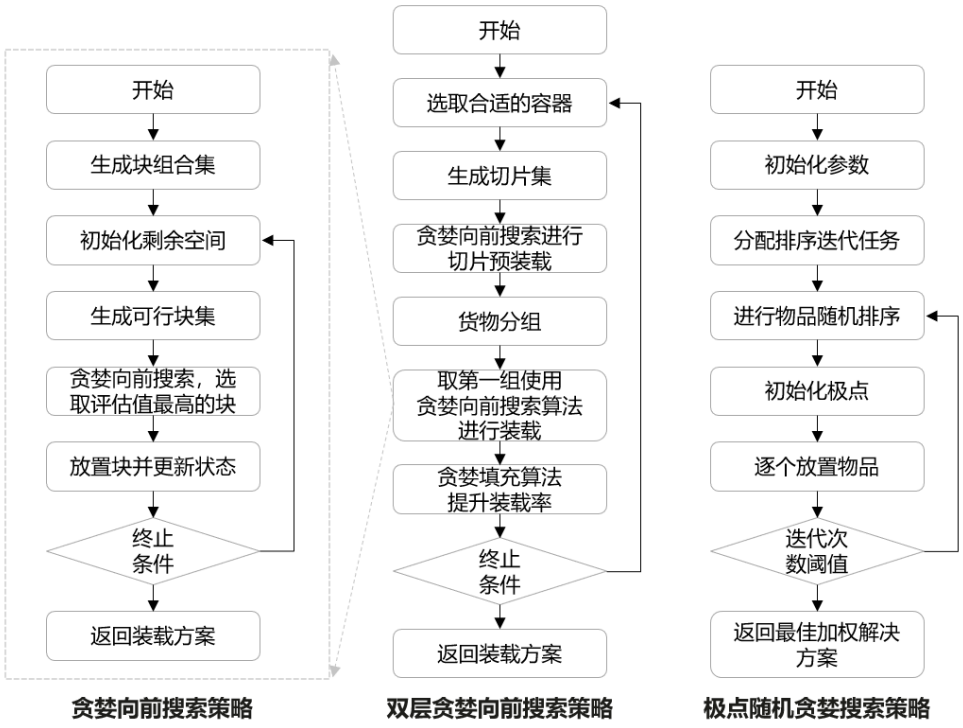

图3.16 三维装箱求解器求解算法流程图

三维装箱求解算法主要包含三类：贪婪向前搜索算法、双层贪婪向前搜索算法和极点随机贪婪搜索算法。图3.16展示了三类算法的求解流程图，详细求解流程详述如下。

1) 贪婪向前搜索算法

贪婪向前搜索算法的主体流程如下：

a) 针对选定的容器，生成所有可能的简单块（见图3.17）组合，并对其按指定规则排序；

b) 初始化当前的剩余空间；

c) 基于给定的剩余空间，生成满足当前装载状态下（即，考虑所有装载约束后）的可行块候选集；

d) 遍历可行块集中排序较前的可行块：放置某一可行块，然后模拟继续多装几个块后的装载情况（即模拟容器装得更满时的状态），最后评估该可行块对装载状态的影响，并从中选取评分最高的可行块；

e) 对选取的块进行放置，并更新装载状态和装载计划；

f) 重复步骤b-e直至所有物品都被装完或是容器已无可装载的空间。

该算法通过模拟“向前看几步”的模拟策略，寻找在给定装载状态下的最佳待装载块，在计算效率和求解质量之间取得了良好平衡，具体优势体现在：1）有限前瞻：通过模拟多步装载状态（通常2-3步），避免纯粹贪婪算法的短视缺陷；2）可控复杂度：相比全局最优搜索，计算复杂度大幅降低，适合实际业务场景的实时性要求。

2) 双层贪婪向前搜索算法

考虑分组的双层贪婪向前搜索算法的主体流程如下：

a) 若有多种容器的类型，调用容器选取算法，基于货物体及重量预测当前最佳的容器类型；

b) 针对当前剩余的所有货物，生成切片集，并使用贪婪向前搜索的方式进行对切片进行预装载；

c) 基于预装载结果，对货物进行分组；

d) 选取第一组（装载结果最佳的组），使用贪婪向前搜索算法进行装载尝试，并结合贪婪填充算法提升装载率；

e) 重复步骤a-d直至所有物品都装载完毕。

该算法通过引入分组预装载和双层决策机制，在基础贪婪向前搜索之上可进一步满足额外的分组约束，使全局优化能力显著增强：（1）分组预判：通过切片预装载对货物进行智能分组，从全局视角识别最佳组合模式；（2）避免局部最优：有效防止基础算法因单一顺序装载而陷入局部最优解；（3）可适配额外的业务约束，如订单不可拆分在不同容器中装载等规则约束。

3) 极点随机贪婪搜索算法

极点随机贪婪搜索算法的主体流程如下：

a) 针对选定的容器，初始化随机化贪婪搜索（RGS）参数，包括随机化程度、支撑比率、重心偏差等；

b) 为每个排序标准（例如体积优先、随机等）准备迭代任务，每个标准分配一定数量的迭代；

c) 并行执行插入启发式算法：对每个任务排序物品、初始化极端点，然后逐个尝试加载物品到容器中，最后更新极端点和解决方案；

d) 若总迭代次数未达到最小值，继续额外迭代以确保充分探索；

e) 评估每个解的质量，包括体积利用率、重量平衡得分等；

f) 基于综合得分返回针对该容器的最佳解决方案；

g) 重复步骤a-f直到所有物品都被装载完成或是已无可用容器。

该算法通过结合多种排序标准、随机化程度以及并行迭代的贪婪搜索策略，专门针对不规则容器的三维装箱问题进行优化，具体优势体现在：（1）通过多种排序标准（如堆叠体积、随机排序）和随机化参数引入解的多样性，避免陷入局部最优，提高了全局解的质量；（2）采用多线程并行处理多个迭代任务，能高效利用计算资源；（3）综合评估函数平衡体积利用率、重量平衡和剩余物品惩罚，更好地适应实际装箱约束和多目标优化需求；（4）通过关键面、极端点和碰撞检查支持了不规则容器的物品装载。

3.5.5 三维装箱求解器功能特性

常规三维装箱问题通常仅考虑物品尺寸与容器匹配，并确保物品间无重叠。然而，在实际业务场景中，如集装箱配载、货车装运、航空箱装配等，所需满足的约束条件更为复杂，典型应用包括：

1) 多品类货物混合装载；

2) 多类型容器动态选型；

3) 按业务规则进行货物分组装载；

4) 多提货点至单一卸货点的集成配送；

5) 单一提货点至多卸货点的顺序卸货装载；

6) 非标准形状容器（如航空箱）的适配装载。

在上述场景中，系统所支持的装载目标与关键特性可归纳如下表所示。

表3.7 三维装箱求解引擎特性列表

分类	支持功能	说明
优化目标	容器总使用量最少	使用的总容器数量最少
	容器实装率最大	容器实装率最大
	费用最小	按运费最小的原则自动优化配置方案
物理约束	几何约束	要求货物形状为立方体
	尺寸约束	货物总长/宽/高不能超出容器的内长/宽/高
	载重约束	装载重量不超过容器装载上限
	货物方向约束	支持立放、平放、侧放三种基础方向
	堆码层数约束	同一类别的货物堆叠时限制堆码层数
	货物承重约束	1. 根据货物材质分配上下层承重关系，定义“承重级别”，级别高的货物不可放在级别低的货物上面 2. 定义货物必须放置在最底部 3. 定义货物上方不允许装载其他物品 4. 不同底面尺寸的货物在堆叠时总重量不大于箱型承重上限
	悬空比率约束	允许上层货物部分悬空以提升实装率（如悬空比例≤30%）
	重心平衡约束	货物重心偏移量不超过几何重心给定阈值
	容器数量约束	每类容器有给定数量时，使用量不超出给定数量
	缝隙预留约束	为易碎品或加固件预留空隙
	角件规避约束	自动避开集装箱内部角件
	货物限高约束	货物最高不超过固定高度
	门口限高约束	在接近门固定长度内，货物总高度不超过门高
路线相关（如集装柜/货车装载）	路径回环约束	每辆车的访问路线不能存在子回路
	装载顺序约束	从不同仓库发运，或发往不同目的时，每辆车的货物装载次序遵守给定的次序，限定局部混装的空间大小
	便捷卸货约束	免倒箱卸货，即前一个提货点的货物不允许遮挡后一个提货点的货物
	相邻平台货物堆叠深度约束	相邻提/卸货点的箱子在堆叠时，限制装载深度，避免因放置位置太深而无法够到
业务约束	同类聚集约束	实际装柜时，将同一物品尽量摆放一起，不同物品之间允许局部的混合搭配
	优先装载约束	多货装载时，尽量让同一货物装在一个容器中，只要容器达到既定的装载率即可
	完整订单约束	订单不拆单，即一个订单只会被装载在一辆车上，除非提供的最大尺寸车辆也无法一车装完此订单
	物品混装约束	打托或装柜时，要求只允许同一订单或同一目的

		地的物品混装。
	成套装载约束	一款产品分拆为多个箱规，或一套产品由多个不同产品组成，装柜时必须保证这些箱规按比例装载
	先打托后装柜约束	1. 货物全部打托，再装载 2. 部分打托，再和散货一起装载

3.5.6 三维装箱求解器关键技术

三维装箱求解技术是一套面向复杂工业场景的高性能求解引擎，当前业界主要采用近似求解策略，如启发式算法、元启发式算法、机器学习方法及混合算法等，不仅要考虑几何尺寸的基础装箱问题，还要考虑超过20项物理与业务约束，如载重、堆码、重心、订单分组、装卸顺序等，并支持多目标优化，如最少容器、最大实装率等。为平衡求解质量与计算效率，三维装箱算法采用多层级的贪婪搜索框架，融合了贪婪向前搜索、分组双层贪婪搜索及极点随机贪婪搜索三大核心策略，在短时间内即可生成高质量的装载方案，有效替代耗时数小时的人工模拟。三维装箱求解器涉及的关键求解技术如下所述。

3.5.6.1 可行块

在实际物流与仓储场景中，待装载的物品通常可根据其外廓尺寸、允许的放置朝向等属性划分为若干类别，同类物品具有高度相似性。基于此，通过预定义“块”结构并对输入物品进行组合优化，可显著降低三维装箱问题的搜索空间与计算复杂度[136][137]。

“块”是指由多个完全相同尺寸的箱子紧密排列、无空隙组合而成的长方体或正方体结构。一个块在X、Y、Z轴方向上分别包含nx、ny、nz个箱子，箱子总数为$nx \times ny \times nz$。在生成块的过程中，算法可同步考虑箱体的堆叠层数限制、承重约束以及货物的允许放置方向等实际条件，确保生成的块既满足几何组合规则，也符合物理装载要求。如图3.17所示，4个同类箱子可生成13种组合块。通过将多个单一物品聚合为块单位进行统一放置，算法无需对每个物品进行独立位置搜索，从而大幅减少决策变量数量，提升求解效率，尤其适用于大规模、高相似度的装箱场景。

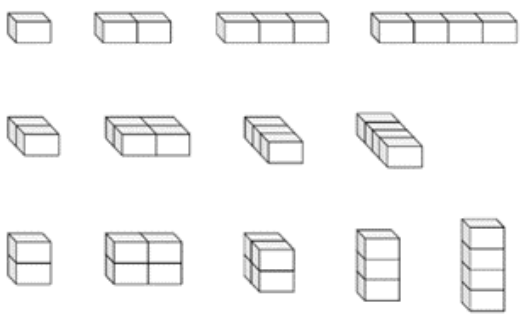

图3.17 三维装箱中“块”的概念示例说明

3.5.6.2 贪婪向前搜索

为克服传统贪婪算法易陷入局部最优的局限性，本算法引入贪婪向前搜索机制，通过有限步长的前瞻性评估来指导块的选择与放置顺序。如图3.18所示，depth表示前向搜索的深度，branch表示前向搜索的宽度，即下一块可从branch个块中进行选择。

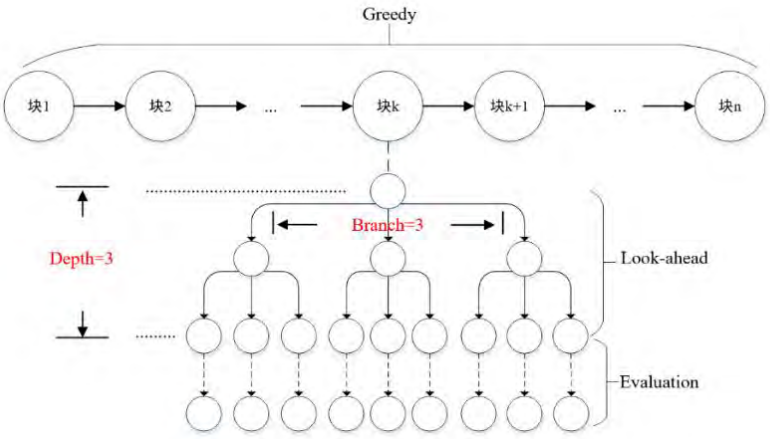

图3.18 三维装箱贪婪向前搜索的概念示例说明

当前向搜索达到预设深度depth时，优先对剩余空间以贪婪策略快速补全，后对当前模拟的装载状态进行整体评分。评价函数综合考虑已装载块的布局紧凑性、空间利用率，从而量化当前搜索路径的潜在价值，用于指导上层选择最优装载方案。

3.5.6.3 极点

在三维装载算法中，“极点”指装载空间内可供放置新物品的候选位置点，其核心价值在于通过系统化的位置枚举提升装载方案的灵活性与空间利用率[138]。极

点通常通过几何投影方法动态生成，例如将现有极点沿坐标轴向容器内壁或已装载物品表面投影，从而衍生出新的候选位置[139]。每完成一次物品装载，算法都会执行投影操作以更新极点集合，确保覆盖所有潜在的有效装载位置，避免遗漏更优解。极点生成与使用的具体流程包含三个关键步骤：

1) 极点初始化：通常以容器内自定义坐标原点（$0, 0, 0$）作为初始极点。若容器为不规则形状（如航空集装箱），原点可能位于容器外部，此时则选择容器内距离原点最近的有效顶点作为初始极点；

2) 确定投影起点：在放置物品后，需从当前极点集合中筛选出合适的点作为投影操作的起点；

3) 投影生成新极点：从选定的起点出发，沿指向容器原点的方向进行投影，直至与容器边界或已装载物品表面相交，该交点即为新生成的极点。

该方法通过考虑物品的间接接触面，能够生成更丰富的候选位置，显著提升空间利用潜力。在处理不可堆叠物品时，算法可跳过那些无法提供直接支撑的极点，确保满足物品的物理特性要求。同时，通过极点接触关系可有效判断物品的支撑状况，防止悬空，保障装载稳定性。此外，极点法天然支持非立方体形状容器（如航空集装箱、异形容器），展现出良好的算法通用性与工程实用价值。

4 华为天筹求解器技术方案与应用实践

本章聚焦华为天筹求解器在千行百业的技术方案与应用实践，精选了航空、钢铁、有色金属、医药、能源、云计算、仿真等17个典型落地案例。每个案例均深入剖析业务背景与核心需求，详细阐述针对性的技术求解方案，并展示应用成效与推广价值。从复杂的航线网络规划、生产计划与排程到大规模电力调度与云资源管理，天筹求解器通过攻克行业痛点，实现了从传统经验决策向智能科学决策的跨越。

本章旨在通过这些真实的实践场景，全方位展示天筹求解器赋能企业降本增效、驱动数字化转型的能力，为相关行业的智能化升级提供可借鉴的参考。同时，本章希望强调，优质项目的落地不仅取决于资金投入，业务流程的清晰梳理、IT系统的完备支撑以及高质量的数据治理同样是项目成功的重要条件。

4.1 航空公司航线网络计划技术方案与应用实践

4.1.1 业务背景介绍

4.1.1.1 业务场景描述

航空公司的航班计划分为年度计划、换季计划及执行计划。其中年度计划为提前编排未来一年全年内的航班计划，换季计划指航季（冬10月-次年3月，夏4月-9月）的航班计划，执行计划为开始售票后3个月之内的短期航班计划。年度计划属于长期计划，是后续换季和短期航班计划的基础，后续航班计划都是基于年度计划调整而得到的，因此高质量的年度计划是航司航班计划编排的重点。

航班收益评估：根据od（Origin-Destination，出发和目的地配对）行业客流，航班吸引力指数，预估航线内航班客流及边贡（收益-成本）。线内航班频率及时刻需匹配航线收益情况，热门航线需增加航班密度，冷门航线可能涉及航班调减，合理的航班编排方案有利于提升航空航司收益。

运营调度与资源分配：航空公司的资源可分为实体及虚拟两部分：其中运力

（飞机）及机组（飞行员）可视作实体资源；时刻（飞机在某机场允许起飞降落的时间点位）、航权（航司在某两个机场间可安排航班的个数）可视为虚拟资源，在有限的资源范围内，编排航班计划，确保航班高效运行。

安全与合规管理：民航总局对于机型可起降机场有严格规定：例如高高原机场仅限指定机型可以起降；各机型在机场有最小保障时间；定检计划频率等，这些都是为了保障飞机及旅客安全。航班计划编排需在严格满足安全和归类管理规定的基础上，提升飞机利用率（轮档时间/天/架）。

行业协同与战略发展：长期计划助力航空公司布局“一带一路”等国家战略航线，重点枢纽机场航班波打造，提升国际竞争力。补贴航线、国际航线、品牌航线是航班计划中不可或缺的部分，必须保障这些航线上的航班正常运行。

总的来说，年度航班计划需在满足资源约束，安全管理规定的前提下，以收益最大化为目标，编排航空公司年度航班计划。在收益计算过程中需结合od客流预测，航班吸引力模型提升航班边贡预估准确性，端到端形成数字化解决方案。

4.1.1.2 业务痛点

1) 航班计划依赖人工经验：基于历史航班计划，依靠人工经验，调整局部航班计划，可能导致客流预估不准确，航线航班频率不合理，飞机利用率不足；

2) 计算效率瓶颈：人工调整单周数万航班计划，缺乏高效计算工具支撑中远期航班计划编排，调整后优化目标变化情况无法及时准确评估；

3) 航班计划单一：航季内航班计划按周重复，无法体现淡旺季航线客流及月度运力差异。

4.1.1.3 技术挑战

1) 数据质量与复杂性：航班计划编排涉及大量及复杂的资源数据，历史数据表分散，关联关系复杂，维度不统一；

2) 约束严格与多样性：航班编排所需满足的约束复杂，涉及时刻、航权、运力、运标及衔接等；在复杂的约束空间中找到全局最优解的难度大；

3) 算法执行效率：单周万级时刻及航权数据，万级航班边贡快速评估，现有方案无法在分钟级完成复杂计算，支撑航班计划高效调整。

4.1.2 业务需求

4.1.2.1 客户整体功能需求

业务需求：

1) 航班计划体现月度运力差异；

2) 航班计划体现月度客流差异；

3) 与历史计划差异可控。

功能点：

1) 航班串周期间衔接修复：修复周与周之间航班串衔接问题；

2) 航班整合：在指定机型范围内对于航班串进行整合，尽可能合并航班串，节省运力；

3) 运力适配（机型映射）：航班串机型适配；

4) 飞机利用率均衡：各航班串轮档时间尽量均衡；

5) 航班串内新增航班：航班串内有时间空隙的位置以边贡最大化目标插入新增航班；

6) 多个航班串混排（时刻再利用）：N个航班串内航班释放，收集时刻、航权、运力等资源后以边贡最大化重新规划航班计划；

7) 机型调整：根据航班边贡，进行航班（往返）机型调整；

8) 增班：已有航线及新增航线增班；

9) 航班波：重点航线航班波及自定义航班波调时；

10) 减班：指定航线及已有航线根据边贡率减班；

11) 航班编排结果可视化：甘特图形式展示航班编排结果及优化目标等；

12) 边贡计算：对接QSI和客流预测结果，获取航班、航线边贡，客座率、飞机利用率。

4.1.2.2 集合和常量

1) 资源数据：

a) 时刻数据，具体到机场、周期、时刻点位、区分起降的时刻个数；

b) 航权数据，具体到起飞、中转、降落机场、周期的班期个数。

2) 运力数据：具体到分子公司、时间范围、机型的可用运力数量。

3) 运标数据：

a) 过站时间：具体到机场、机型的最短过站时间；

b) 航段时间：具体到起飞机场、降落机场、机型的过站时间范围；

c) 机场限制：具体到机场、机型起降的限制；

d) 减载：具体到机场、机型的减载数量。

4) 历史客流票价：

a) 客流：具体到起飞降落机场、日期的客流；

b) 票价：具体到起飞降落机场、日期的票价。

5) 成本数据：具体到起飞降落机场、机型的变动成本数据。

4.1.2.3 业务目标

优化目标：航班计划全局边贡最大化、飞机利用率最大化、最小化运力使用。

航班结果输出：支持输出周、月度及年度航班计划；支持输出航班边贡、客座率及飞机利用率等统计结果；支持输出航班计划甘特图等可视化界面。

4.1.2.4 业务约束

1) 时刻约束：航空公司单日在某机场的某个时刻（12:00）最多可以起飞/降落N个国内/国际航班；

2) 航权约束：航空公司单周在起飞城市-中转城市-到达城市间最多可起降N个往返航班；

3) 运力约束：航空公司每月每执管公司每营销机型最多可使用的飞机数量；

4) 飞行时间约束：航空公司全年每机型大类总可飞行时间（飞行员数量限制）；

5) 运行标准：营销机型可以起降的机场；

6) 特殊机场：高高原机场机型限制；

7) 航段时间：两机场间最小/最大飞行时间；

8) 过站时间：每机场每营销机型最小过站时间（航班间最小时间间隔）；

9) ETOPS时间：洲际宽体机航班最小保障时间；

10) 固定航班：国际、品牌、补贴及人工指定等航班需保持不变（起降机场、起降时刻及机型）；

11) 业载：每机型，起飞降落机场最大业载；

12) 飞机利用率：以边贡最大化为目标时，飞机利用率按机型不低于x；

13) 航班机型稳定：同一个航班号的航班尽可能分配同一营销机型；

14) 航班结构稳定：航班尽可能按往返编排，即A-B-A或A-B-C-B-A,尽量减少甩边单飞。

4.1.3 技术方案

考虑客户模型约束复杂且严格，模型规模大，设计定制化优化算法“考虑航班时刻协调决策的航班串机型匹配模型”[140]，结合业务，集合航班粒度，降低模型规模及求解难度，通过混合算法实现高效搜索方案。

基于业务需求，考虑航班优化场景约束严格且复杂，优化目标评估难度高，结合时序大模型进行客流预估，多因子模型进行航班吸引力指数预估，高精度计算航班客流，基于天筹求解器和优化算法对模型进行高效求解。该方案旨在为航空公司在航季、年度，提供“航线网络规划”算法服务，解决客流预测不准确、航班吸引力模糊、航线网络规划优度低等问题。

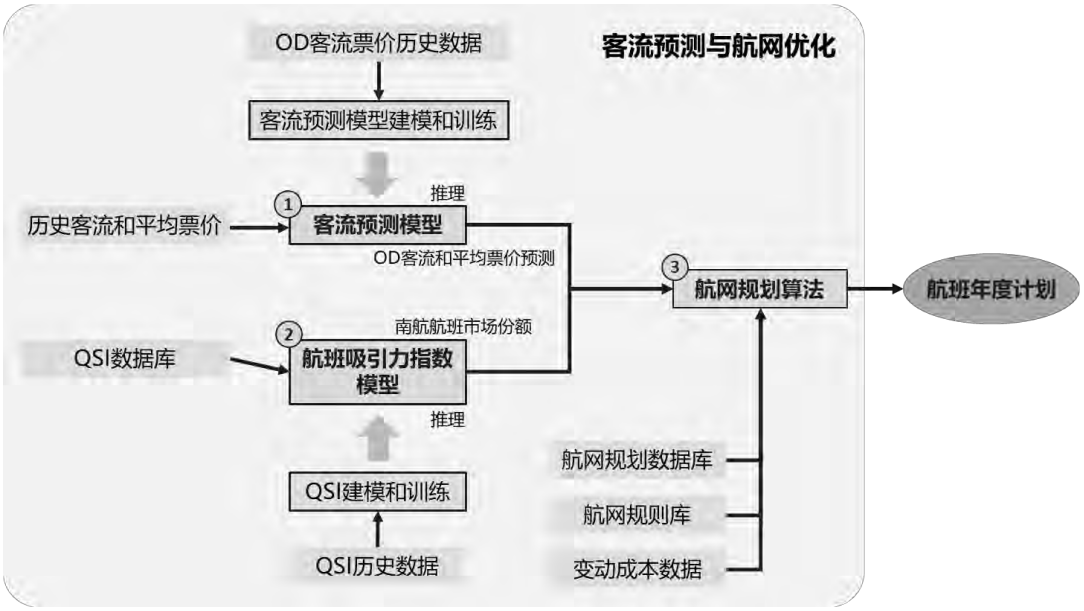

图4.1 航司年度航班计划整体解决方案

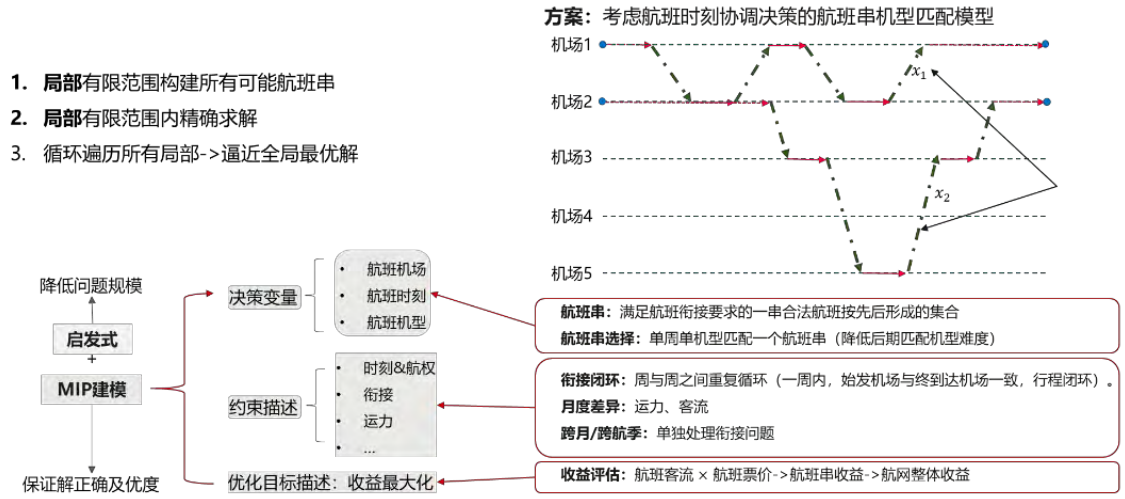

图4.2 航网优化场景解决方案

4.1.4 应用成效

1) 高效求解：分钟级求解，相对最优解，提升边际贡献；

2) 功能全面：增班，减班，运力优化，机型优化，航班波构建，时刻再利用；

3) 时刻再利用局部资源内，航班边贡最大化全局最优解；

4) 航班计划人工接纳率>65%。

4.1.5 复制推广建议

该航班计划编排方案为全流程航班编排，包括客流预测、航班吸引力分析、航班编排，客流预测部分对于客户的数据要求较高，因此对于规模不同的航司可区分推广方式：

1) 大型航司：对于航班编排要求较高，可影响地区航班量，建议完整方案推广；

2) 腰部航司：资源及数据有限，可单独推广航网编排部分，前置客流、票价等用历史数据做替换。

航班调整功能丰富，不局限于年度航班计划，亦可用于换季计划，季中调节等部分。

4.2 轮胎生产排程计划技术方案与应用实践

4.2.1 业务背景介绍

4.2.1.1 业务场景描述

轮胎生产属于典型的橡胶加工+部件成型+硫化工艺，核心是先把各种原材料加工成不同“半成品部件”，再把部件组合成胎坯，最后高温高压硫化成成品轮胎。关键生产环节如下（图4.3和图4.4分别为轮胎部件构成示意和生产流程示意）：

1) 部件生产环节：针对上游混炼好的各种胶料，通过挤出、压延、裁断、贴合等工序，分别制成胎面、胎侧、内衬层、帘布层、带束层、胎圈等20+余种半成品，供成型工序组装成胎坯；

2) 成型生产环节：将半部件在成型机上按工艺顺序依次贴合、滚压、反包、组合，形成结构完整、尺寸精确的胎坯的过程，是连接半部件生产与硫化的关键工序；

3) 硫化生产环节：是轮胎制造的最终定型工序，将成型后的胎坯送入硫化机，在模具内通过高温、高压、特定时长的协同作用，使橡胶发生交联反应，实现轮胎花纹定型、结构固化，形成成品胎，是连接胎坯与合格成品的关键工序；

4) 成品检验环节：检验是轮胎出厂前的最终质量把关工序，指通过外观筛查、尺寸测量、性能测试、无损探伤等多维度检测手段，验证成品胎是否符合产品标准，剔除不合格品，是连接生产端与市场端的关键质量管控节点。

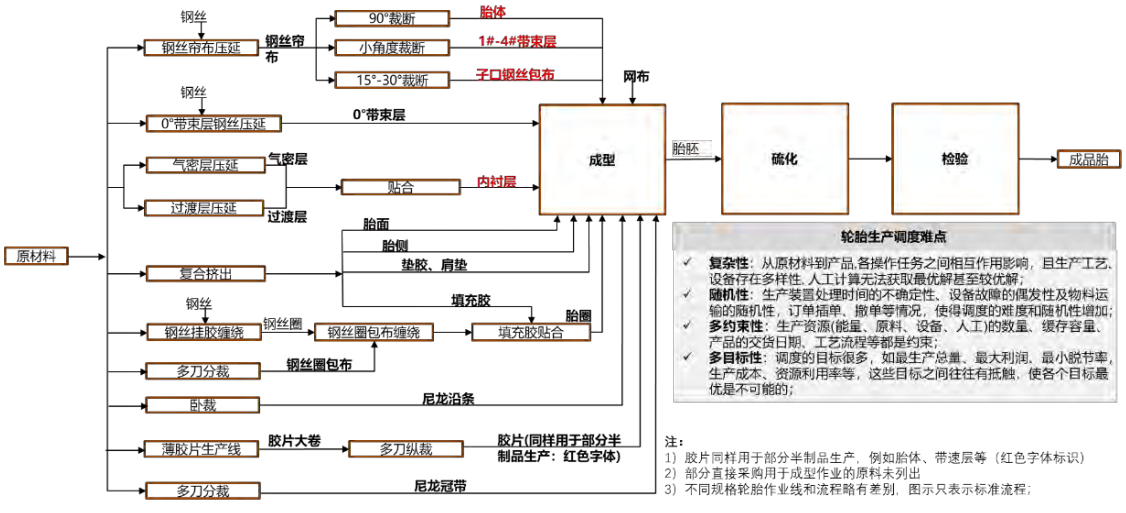

图4.3 轮胎部件构成示意图

图4.4 轮胎生产流程示意图

其中，硫化/成型机台作为贵重生产设备，属于生产瓶颈资源；同时，成型与硫化是轮胎制造中前后衔接、相互制约、深度耦合的核心工序，涉及生产节拍同步要求、换型协同要求、换模协同要求、胎胚停放要求等；二者的协同效率直接决定了轮胎产能、质量与成本[141][142]。

4.2.1.2 业务痛点

当前轮胎生产场景存在如下痛点：

1) 订单交期管理缺乏精准性：无法通过科学方法确定准确交期，可能导致客户满意度下降或订单履约风险；

2) 设备利用率偏低：资源配置缺乏优化，成型机、硫化机等强关连生产设备协同不足，未能充分发挥产能，造成资源浪费；

3) 人工操作负担重且灵活性差：依赖人工进行编码透视、系列分类等基础工作，不仅工作量巨大，且面对订单变化、规格调整等情况时，难以快速响应调整；

4) 排产结果缺乏量化评估：没有明确的量化指标衡量排产计划的优劣，无法有效判断计划对成本、效率、交期等核心目标的支撑程度。

4.2.1.3 技术挑战

当前轮胎生产场景存在如下技术挑战：

1) 节拍差异与产能平衡挑战：成型工序单胎产出节拍远快于硫化节拍，且硫化节拍随轮胎规格、胶料配方差异显著，易出现“成型胎坯堆积”或“硫化机闲置”的产能失衡；

2) 资源约束与调度复杂度挑战：硫化模具成本高、数量有限，且不同规格轮胎需专用模具，排产时需优先保障高优先级订单的模具占用，同时避免模具频繁切换导致的效率损耗；

3) 动态扰动与响应灵活性挑战：紧急订单插入、常规订单变更需快速调整排产计划，而成型-硫化的联动性导致“牵一发而动全身”，调整成型排产需同步更新硫化机负载，反之亦然，易引发连锁反应；

4) 多目标优化与约束平衡挑战：为提升硫化机利用率，可能会集中生产同一规格轮胎，但会导致成型胎坯库存积压；但若按订单顺序小批量生产，又会增加换型/换模成本；排产需在“库存成本”、“换型/模成本”、“订单履约率”等多目标间找到最优解。

4.2.2 业务需求

4.2.2.1 客户整体功能需求

针对成型-硫化场景，传统排产依赖人工经验，面对每月近1500+个订单、30万条胎、近 400 种轮胎规格、25 台成型机、190 余台硫化机及 400 余种模具构成的千万级规模复杂场景；客户希望通过算法赋能，综合平衡产能利用率、客户交期、交货量等目标，以图4.5的排产需求示意为例，最终输出以下三个计划：

1) 成型月度作业计划：未来一个月每台成型机每个工作日生产的胎胚规格及对应产量；

2) 硫化月度作业计划：未来一个月每台硫化机每个工作日生产的成品胎规格及对应产量；

3) 硫化3日换膜计划：未来3天每台硫化机每个工作日使用的模具型号及数量。

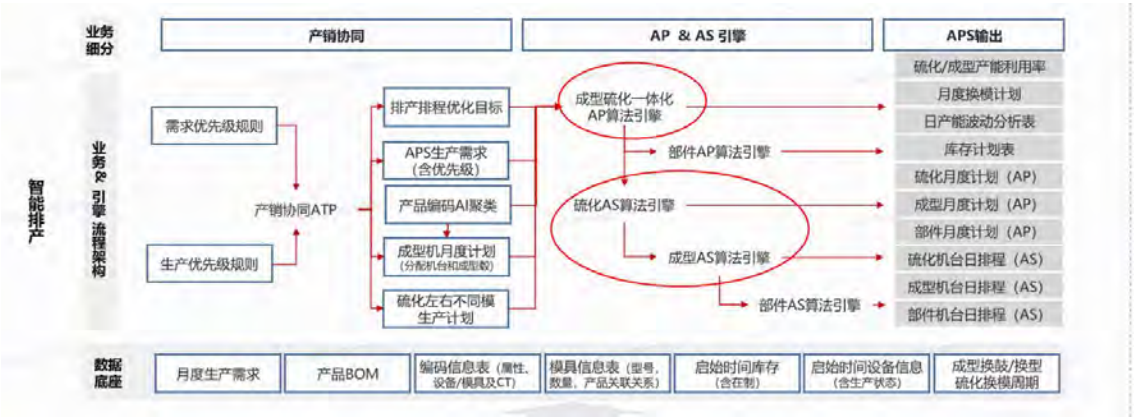

图4.5 轮胎排产需求示意图

4.2.2.2 集合和常量

轮胎排产场景涉及的关键数据集合和常量如下：

1) 硫化机信息：机型、硫化机编号、模位数、是否同起同落、可作业成品胎编码、当前配置可生产的成品胎编码、开合模时间、最大合模力、可安装模套类型、检修计划；

2) 成型机信息：成型机编号、成型机型、可作业胎胚编码、可作业胎胚编码的单胎周期、当前配置可生产的胎胚编码、检修计划；

3) 模具信息：模具类型、可用数量、可作业成品胎编码；

4) 物料清单信息：成品胎编码、胎胚编码、胎面编码、胎侧编码、内衬层编码、肩垫编码、胎圈编码、胎体帘布、零度带束层、趾口加强层、尼龙沿条、尼龙冠带、胎体尼龙沿条、趾口筛网布等；

5) 工艺要求信息：外胎规格、胎宽、寸别、花纹规格、商标、槽宽规格、合模力、层级、内胎、使用范围；

6) 要货信息：成品胎编码、需求数量、交货期限、作业优先级；

7) 转产耗时：硫化换模耗时、成型换型耗时、硫化换规格耗时。

4.2.2.3 业务目标

表4.1为轮胎排产场景的业务目标概述。

表4.1 轮胎排产业务目标

目标	描述

需求量满足	每个成品胎编码有实际需求量；排产结果应尽量满足编码的需求量
交期满足	每个订单都有交期要求，排产结果应尽量满足给定交期；
优先级满足	每个订单有排产优先级要求，优先级高的要尽量保证交期和数量；
最小化硫化换模次数	单次换模耗时较高（>400min），因此需要减少硫化换模次数
最小化成型换型次数	单次换型耗时较高（>30min），因此需要减少成型换型次数
最小化胎胚库存	成型区和硫化区之间存在胎胚的库存，在满足成型、硫化产能前提下，胎胚库存越小越好；

4.2.2.4 业务约束

表4.2和表4.3分别为成型机和硫化机的生产约束。

表4.2 成型机排产业务约束

序号	描述
1	成型机-胎胚可达约束
2	成型机换毂次数约束
3	成型机台产能约束
4	成型毂数量限制
5	撑块数量限制
6	连续生产约束
7	生产节奏约束
8	成型机换型、换毂耗时计算规则
9	排产量锁定约束
10	成型机检修计划时间约束
11	成型机单机台单日生产胎胚规格数上限约束
12	成型单日生产胎胚规格数上限约束
13	成型组合黑名单约束
14	胎胚与成型毂的可达关系
15	成型毂-成型机台可达关系约束

表4.3 硫化机排产业务约束

序号	描述
1	胎胚最长停放时长约束
2	硫化机-成品胎可达约束
3	硫化机台产能约束
4	成套模具数量限制
5	成品胎-成套模具有可达约束
6	硫化机检修计划时间约束
7	连续生产约束
8	模具清洗约束
9	生产节奏约束
10	硫化机换膜耗时计算规则
11	硫化机双模是否同起同落作业约束
12	硫化机左右不同模（拼模）作业人工规则

13	硫化AP结果推演小时级换膜计划规则
14	每日换模次数上限约束
15	订单交期应答要求
16	拼模场景下的可超产量限制约束
17	异常模具替换规则

4.2.3 技术方案

基于混合整数规划方法，构建完整的数学模型体系，图4.6为轮胎排产技术方案示意图，具体实现路径如下：

1) 模型构建：明确决策变量（如各设备生产批次、切换时间节点等），建立涵盖资源约束、切换约束、产能约束、损耗约束的多维度约束条件，同时以“提升生产效率与轮胎产量、减少胎胚库存浪费” 为核心目标，构建目标函数；

2) 模型求解：借助天筹 AI 求解器等专业求解工具，对构建的数学模型进行高效求解，确保求解结果的最优性与时效性；

3) 方案输出：根据求解结果，生成符合实际业务需求的精细化排产方案，明确各设备、各批次的生产计划、切换安排等关键信息。

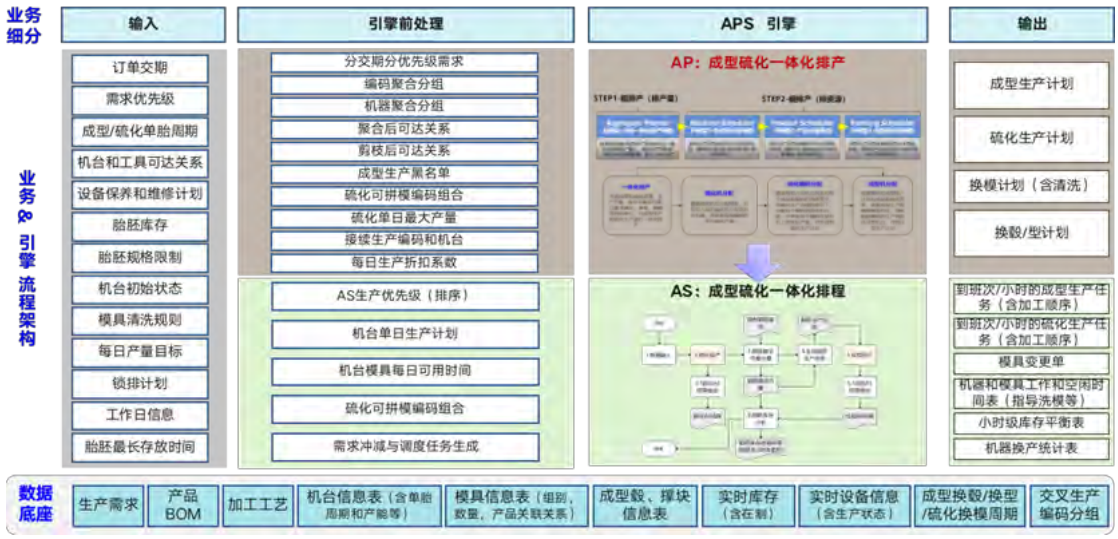

图4.6 轮胎排产场景解决方案

4.2.4 应用成效

通过上述技术方案，在满足业务需求的同时，同步有如下应用成效：

1) 生产效率提升：通过科学调度化解资源冲突、减少切换损耗，最大化发挥成型机、硫化机的产能潜力，提升单位时间生产效率，换型次数减少5%；

2) 产量精准保障：以月度产量计算为核心目标，通过精准的约束控制与目标优化，确保月度产量目标的达成，三日换模计划执行率达95%；

3) 库存成本降低：通过优化胎胚生产与硫化的衔接节奏，减少胎胚库存积压，降低库存占用成本与浪费风险，胚胎库存减少10%；

4) 决策科学性增强：为生产调度提供数据驱动的科学排产方案，替代传统经验型调度，提升生产决策的精准度与可靠性，单次排产计算耗时从天级降到小时级。

4.2.5 复制推广建议

具有成型-硫化工艺要求的橡胶制品生产排程场景，核心应用领域包括：

1) 汽车工业：适配轮胎、减震器、密封圈、胶管、减震垫等耐磨、耐高温橡胶制品的生产排程。汽车工业对橡胶制品的工艺精度、交付时效要求极高，且产品型号繁杂，可结合成型-硫化的设备产能、工艺节拍、订单优先级，优化生产序列，解决传统排程中“设备闲置、订单延误、工艺参数不匹配”等问题，保障批量生产与定制化需求的高效平衡；

2) 工业机械：涉及各种密封件、胶辊、传动带、弹性减震元件的生产。该领域产品多为定制化生产，订单批量小、规格多，生产排程易出现“切换频繁、效率低下”的问题，可根据订单规格、工艺要求、设备负荷优化生产顺序，减少设备换模时间，提升单位时间产能，适配机械制造企业的柔性生产需求；

3) 建筑与工程：覆盖防水卷材、隔音垫、门窗密封条、耐老化组件的生产排程。建筑用橡胶制品多为大批量生产，且受项目交付周期约束，传统排程依赖人工经验易导致“产能浪费或交付滞后”，可结合成型-硫化的生产周期、原料库存、交付节点，优化排程方案，确保批量生产的高效推进与交付时效达标；

4) 电力电子：适配绝缘子、电缆护套、电子配件密封等橡胶制品的生产场景。该领域产品对绝缘性、耐候性等工艺指标要求严苛，生产过程中需严格控制硫化温度、时间等参数，可将工艺标准嵌入排程算法，确保生产过程中设备参数、工艺要求与订单需求精准匹配，减少不合格品率，提升生产稳定性；

5) 医疗与家用：包括医疗器械密封件、医用胶塞、手术手套、厨房厨具胶件等产品。其中，医疗类产品需符合严格的卫生标准，家用产品则注重性价比与产能效率，可根据订单批量优化生产节奏，平衡合规性与生产效率，适配医疗企业与家居

用品制造商的双重需求；

6) 鞋业与橡胶板：适配鞋底制造、防震胶板、橡胶地垫等产品的生产场景。该领域产品生产批量大、工艺相对标准化，但易受原料供应、设备产能波动影响，传统排程易出现“供需错配”，可结合设备利用率、原料库存水平、订单量变化，实时优化排程方案，减少原料积压与设备闲置，提升整体生产周转效率。

4.3 医药供应链计划流程优化技术方案与应用实践

4.3.1 业务背景介绍

4.3.1.1 场景描述

中医药供应链计划的整体流程是一个涵盖中药材全生命周期的复杂系统，其核心特征体现为“源头把控-工艺传承-合规流通-动态响应”的闭环管理。该模式以道地药材资源为基础，通过“种植/采集-采购检验-炮制加工-制剂生产-仓储物流-终端销售”六大核心环节的协同运作，构建起符合中医药传承创新需求的供应链体系。

在医药行业备货式生产中，企业需基于药品销售预测提前规划生产批次与库存策略，以应对市场需求波动。然而，这一模式需同时应对中药材生长周期长、产地气候敏感、炮制工艺复杂等自然属性，以及药品监管政策动态调整、市场需求快速变化等商业挑战，要求供应链计划系统具备多维度数据融合能力、动态预测模型迭代能力及跨部门协同响应能力。

中医药的行业特性导致场景复杂度极高[143]，其供应链各个阶段行业特点总结如图4.7所示：

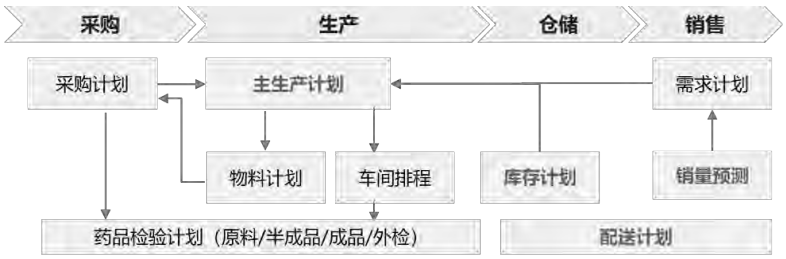

图4.7 中医药供应链主要环节示意图

1) 采购环节：根据生产需求与市场行情，采购道地中药材（野生采集或人工种

植），需严格筛选供应商并检测药材质量（如有效成分含量、农残指标）。辅料与包装材料采购同样需要符合药典标准；

2) 生产环节：独有的炮制加工阶段按传统工艺或现代技术对中药材进行炮制，提升药效并降低毒性。制剂生产将炮制后的药材制成丸剂、散剂、口服液等剂型，需严格遵循工艺参数；

3) 运输环节：仓储管理分类存储不同药材（如阴凉库、冷藏库），定期检查温湿度并记录，防止药材霉变或有效成分流失。物流配送采用温控运输设备（如冷藏车）配送至医院、药店或电商平台仓库，确保运输过程符合药品GSP要求；

4) 销售环节：通过医院、连锁药店、电商平台等渠道销售，需结合市场需求动态调整库存与促销策略，建立药品追溯系统，记录从采购到销售的全流程信息，确保符合药品监管要求。

整体场景需在质量优先、动态政策响应、供应链不确定性及多层级协同中，实现预测精准化、库存精益化与生产计划最优化，这对算法效率与业务规则适配能力提出极高要求。

4.3.1.2 业务痛点

中医药供应链领域的核心业务痛点包括：

1) 计划标准难统一：因缺少智能化的计划工具，依赖人工经验的传统方法难以保证计划一致性，通过设定较高的安全库存以减少缺断货风险，资金占用严重造成巨大的运营成本压力；

2) 计划效率瓶颈：计划员需要手动处理数千SKU，缺乏高效计算工具支撑多月份长期计划推演；

3) 多层级协同困难：多生产基地、多物料层级协同困难，父子项物料配比失衡，齐套性差。药材标准切换、产能约束、库存限制等多重约束条件下的最优计划制定复杂度高；

4) 品类管理策略复杂：缺少基于精细化品类画像的计划策略，难以执行精准的资源分配。

4.3.1.3 技术挑战

1) 数据质量与复杂性：药品销售数据存在多维度（SKU、SPU、标准、包装规格）、多层级的复杂性，历史数据质量参差不齐，传统清洗方法难以应对药品行业的特殊数据特征；

2) 预测精度与业务适配：药品需求受多种复杂因素影响，传统预测算法难以精准捕捉药品行业的独特业务逻辑；

3) 多级约束平衡：如图4.8的多层级供应网络结构实例图所示，计划系统面临多层级BOM约束、多基地产能约束、生产周期约束、原材料供应周期约束、库容约束等多重约束条件，在复杂的约束空间中找到全局最优解的难度大；

4) 算法执行效率：面对数千SKU、中长期滚动预测的需求，传统算法框架难以在可接受时间内完成复杂优化计算，无法支撑业务所需的快速响应和实时决策。

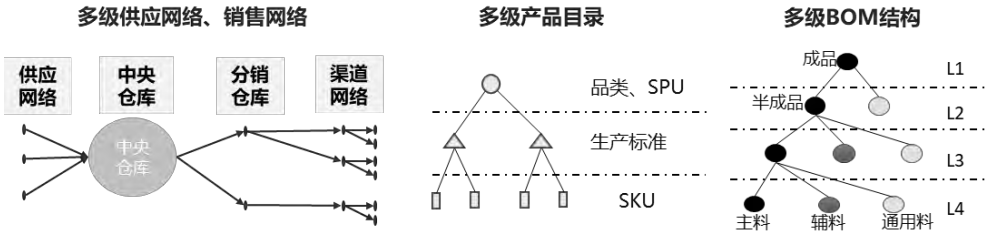

图4.8 多层级供应网络结构示意图

4.3.2 业务需求

4.3.2.1 客户整体功能需求

针对中医药供应链场景，传统计划依赖人工经验，面对药材全生命周期管控、六大核心环节协同运作、数千SKU计划管理和商业环境变化的复杂场景；客户希望通过算法赋能，综合平衡库存周转率、订单满足率、多层级协同效率、政策响应速度等目标，最终输出以下三个计划：

1) 需求预测计划：基于历史销售数据、政策环境、季节特征等多维度因素的中长期滚动预测计划；

2) 库存优化计划：考虑多级仓储备货策略、库存成本控制、药品效期管理的多级库存优化计划；

3) 主生产计划：基于产能约束、工艺要求、原材料供应的主生产计划和排产计划。

4.3.2.2 集合和常量

中医药供应链场景涉及的关键数据集合和常量如下：

1) 需求预测相关信息：

a) 任务配置信息：包含预测周期、预测对象范围等信息；

b) 基础数据：需求分类信息、品种特性信息；

c) 实时数据：月度需求历史数据、库存明细信息、标准切换信息等。

2) 计划算法相关信息：

a) 算法参数配置：当前计划时间，计划展望周期，计划范围等配置信息；

b) 基础数据：物料基础信息、BOM清单信息、生产限制信息、基础设备产能信息等；

c) 实时数据：采购限制信息、库存状态信息、在制在途信息、转运关系信息、产能和工厂日历信息、标准切换信息。

4.3.2.3 业务目标需求

场景主要业务目标和功能需求在表4.4列出。

表4.4 中医药供应链场景业务目标

目标需求	说明
优化目标	1. 最小不满足需求量、药品过期数量； 2. 最小化生产成本和库存成本； 3. 最大化满足安全库存水位。
统计结果输出	1. 支持输出销量预测结果和分位点预测； 2. 支持不同层级物料的最小和最大库存水位建议； 3. 支持计划量、库存量等计划结果相关输出； 4. 支持输出过期、缺货等异常统计； 5. 支持输出生产品类数量、产能利用等统计结果。

4.3.2.4 业务场景约束

表4.5描述了具体的业务场景约束，同时也是中医药行业的典型需求。

表4.5 中医药供应链场景业务约束

场景约束	说明
生产标准切换和限制	1. 新标生效后不可再生产旧标准产品，只允许销售库存； 2. 新标准生产需发布标准后执行。
生产月份限制	3. 指定不同物料的必须满足的生产月份范围。
物料有效期	4. 每个物料或者成品的有效期，如超过有效期视为报废处理。
生产时间窗限制	5. 部分特殊品类物料尽管未过期，但要求必须在加工时间窗内进行下一道工序加工。
加工提前期	6. 包括物料加工周期、转运、检验周期。
最大物料库存限制	7. 不同类型物料的总库存占用有最大限制。
已锁定生产量	8. 已经锁定的工单或者计划量，必须固定在本期计划内的生产量。
建议安全/最大库存	9. 尽可能满足持有库存不低于建议安全库存水位。
生产批量	10. 具有参考的固化生产批量或批量范围，适应生产设备。
瓶颈工序产能	11. 考虑瓶颈工序最大产能限制。
采购月份限制	12. 指定不同品类必须满足的采购月份执行范围。
建议采购月份	13. 支持设置建议采购或补货的偏好月份，涉及药材产新季。
批次生产限制	14. 原料加工阶段尽可能减少尾批剩余，增加生产效率； 15. 满足追溯要求，不可混批生产，拆批必须满足最小生产量。
工艺路线选择	16. 工艺路线受到不同批次生产量影响。

4.3.3 技术方案

4.3.3.1 技术架构设计

如图 4.9 所示，基于天筹求解器构建智能供应链计划一体化技术架构，深度融合需求预测、库存优化、主生产计划三大核心模块，实现数据流、算法流、业务流的三流合一，形成端到端的智能化决策支撑体系。

图4.9 中医药供应链场景整体技术架构

4.3.3.2 关键技术模块

1) 智能需求预测引擎：基于深度学习构建多维预测模型，实现模型池融合预测，支撑多种场景的SKU粒度精准销量预测；

2) 动态库存优化算法：基于马尔可夫决策过程构建多级库存优化模型，采用蒙特卡洛仿真[144]与随机优化方法计算库存策略，实现服务水平保障前提下的库存成本最优；

3) 智能生产计划引擎：通过多目标优化方法处理复杂的计划问题。实现工艺段产能归一化处理、锁定工单智能分配、固化批量动态调整等创新算法，确保在多重复杂条件下的最优计划生成；

4) 高性能计算架构：基于天筹求解器构建大规模并行计算框架，实现算法模块化、策略可插拔、约束条件参数化配置，确保算法的高可用性和可扩展性。

4.3.3.3 方案创新

在需求预测模块中，主要算法融合创新如图4.10所示，主要方法包含：

1) 多模型集成预测：基于场景-模型库匹配机制，通过多种集成技术融合，实现不同预测模型的优势互补；基于上下协同模型，对多标准多规格和多层BOM层级进行协同预测优化，提升预测精度和稳定性。

2) 多维特征工程：构建ABC/XYZ分类特征、周期性动态特征、政策医院等级外部特征的综合性特征体系，深度挖掘药品销售数据的多维度业务价值。

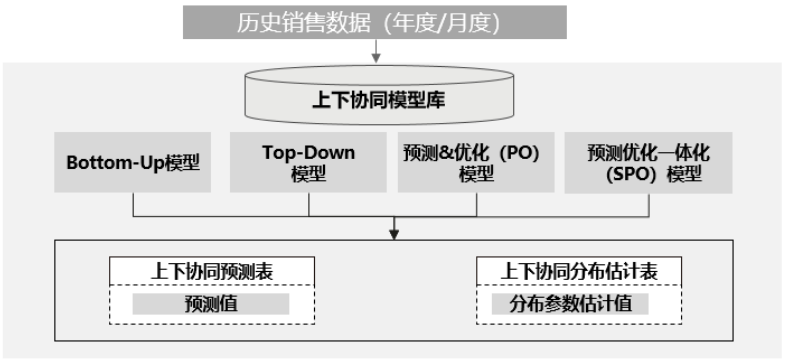

图4.10 需求预测集成技术架构

库存计划中对库存管理理论和算法的创新包含：

1) 策略生成方法论：创新提出"以策略为主、差异化调整"的库存策略生成理论，基于需求特征进行双维度分类，实现对传统策略的有效扩展和定制化优化。

2) 上下游协同算法：首次提出基于卷积库存和卷积订货点的库存校正算法，实现上游库存提前布局，保障成品供应能力的协同优化机制。

生产计划技术创新主要包括：

1) 多目标协同优化：基于天筹求解器构建APS引擎的数学建模创新，通过最小延期量、最小化库存量、最小过期数量等多目标优化方法，处理复杂的计划决策问题。

2) 业务场景适配：创新提出全周期产能平衡、锁定工单智能分配、批量动态调整等算法，确保复杂条件下的最优计划生成。

4.3.4 应用成效

基于应用数据测算分析，量化结果如下：

1) 库存周转效率大幅提升：基于年度周期场景测算，在需求满足率整体达到95%的条件下，库存周转天数减少10%，有效释放了企业现金流；

2) 计划效率突破：算法执行效率实现质的飞跃，从原有的计划员手动处理千种SKU、耗时数日的计划制定过程，缩短至分钟级自动完成；

3) 差异化策略精准落地：实现了上下游库存的协同优化。典型案例显示，半成品库存量可减少20%，在维持服务水平不变的情况下，大幅提升了关键物料的周转效率；

4) 运营成本全面优化：为企业实现从人工经验驱动向数据智能驱动的供应链管理模式转型提供了坚实的技术支撑，大幅提升了供应链的综合运营效率和成本控制能力。

4.3.5 复制推广建议

4.3.5.1 生物制药行业

适用场景：生物药（如单克隆抗体、疫苗）需经历细胞培养、纯化、冻干等复杂工艺，生产周期长达数月，并需符合多项国际监管要求，原料（如细胞株、培养

基）需通过严格质量验证。而疫苗、血液制品等需根据季节性需求或突发公共卫生事件提前规划产能，需要通过库存策略平衡保质期与供应稳定性。

4.3.5.2 金属制造行业

适用场景：高标准化的固件（螺栓、螺母）、轴承、齿轮等，或者管材、线材、板材等金属制品，需求量大且通用性强。作为汽车、机械、电子等交付周期敏感的行业供应商，通常需保持合理的安全库存。此外部分小批量订单组成的长尾需求需通过备货满足紧急需求，此外季节性需求，如空调铜管在夏季需求激增，同样需提前备货。在金属部件制造企业中，定制化与标准化并存，如不锈钢管材需按客户要求切割，但基础管材可标准化生产，对不同层级物料的安全库存策略要求变得更高。因此在该行业场景中，预测驱动补货和柔性生产计划的方案通过，通过精细化管理不同属性和工艺层级的物料，在工厂、区域仓、经销商之间建立多级库存联动机制，最大化订单满足和响应及时率。

4.3.5.3 有机农产品、功能性食品加工行业

适用场景：有机农产品、功能性食品（如益生菌、膳食补充剂）通常都需要较长的转换周期，严格生产工艺以及行业质量标准。如有机农产品需符合有机认证（如USDA、EU Organic），功能性成分含量检测（如益生菌活菌数）及食品接触材料安全标准。在备货式生产的基础上，还需要针对季节性的消费高峰（如节假日、健康季）提前备货，库存需应对电商促销与渠道商订单波动。因此需求预测与库存计划等方案在高峰销量预测和备货策略中的应用至关重要，通过多渠道库存协同和智能补货算法，有助企业平衡线上预售与线下渠道库存，降低滞销风险。

4.4 钢铁板材生产智能板坯计划技术方案与应用实践

4.4.1 业务背景介绍

背景：板坯计划自动组板流程涉及多个转炉和产线；包括三种场景，连铸出钢材组板、余材板坯组板、余材钢板组板。（1）连铸出钢材组板：将已知订单，在指定的结晶器断面进行组板，用于连铸坯料的加工长度的计划，以及指明合同、轧

制板、板坯的对应关系；（2）余材板坯组板（简称余坯组板）：将已知订单，利用指定的库存钢坯进行组板，用于指明订单、轧制板、板坯的对应关系；（3）余材钢板组板（简称余板组板）：将已知订单，利用指定的库存钢板进行组板，用于指明订单、轧制板的对应关系。

痛点：余材板坯、余材钢板组板需人工逐块抽取合同，人工筛选合同组板，反复尝试效率极低；连铸出钢材组板场景，目前只针对简单型A1、A2（类型详情下文介绍）有自动组板，对复杂型S1、S2、S3、S6、G等（类型详情下文介绍）组板类型仍需人工抽取合同、人工筛选组板，同样存在效率极低的情况；影响工作效率，急需进行升级改造。其中板坯计划，特别是余坯组板，为当前的业务瓶颈。

4.4.2 业务需求

组板业务主要分三部分，连铸出钢材组板、余坯组板、以及余板组板。（1）连铸出钢材场景，首先生成虚拟板坯，再组板下发虚拟炼钢连铸生产计划，再匹配相应的实物板坯，该部分功能在随着订单结构发生变化（接单规格小型化、复杂化），复杂组板比例逐年上升，人工组板耗时高，已逐渐无法完全满足业务诉求。（2）余坯组板场景，首先选定库存中的余材板坯，再匹配相应的合同，余坯组板完全由人工进行判断，估算坯料可用范围，并根据交货期选择订单进行组板消化，无法达到坯料利用率的最优模式。如果极致追求坯料利用率，人工需要花费大量时间逐步试错，效率极低。大部分情况下，人工在有限的时间内，无法追求利用率最优。根据历史经验，约40万吨余坯，平均设计成材率为81%。（3）余板组板场景，和余坯场景类似，也是选定库存中的余材板材，再匹配响应的合同，大部分情况下，仅作为单块余板的最近似匹配，人工很难做到组合匹配。

人工现状：计划员认为余坯组板成材率按照以往经验可以最高提升到87%左右，因为人工操作时间有限，因此没有在人工组板方面投入更多精力去匹配重组。通常在有限的时间内，满足一定要求即可(如成材率达到90%以上)，所以基于当前现状，预计有2-5%左右的优化提升空间。

人工操作余材组板流程：（1）先选余材板坯(受板坯库库容影响，部分余材板坯需及时消化)；（2）匹配相应的合同，人工主要判断是否会产生大量浪费；（3）对于余材板坯组板，通过体积不变原则测算预估轧制成品长度，基于展宽比判断轧

制稳定性，最佳展宽比是1.4,稳定性最好；对于余材钢板组板，通过收得率最大化、后道工序加工复杂度最小化进行组板。（4）确定轧制模式以及组板方式(A1、A2类型)。

人工组板存在的问题：（1）效率慢，需要人工逐块按坯料信息从订单中挑选订单组板，影响工作效率；（2）余坯、余板组板利用率低。

规格约束：存在诸多约束条件需要在组板过程中遵守，包括钢种与化学成分类约束、加工尺寸规格类约束、公差类约束、余材处理方式类约束等等，具体如下：

1) 化学成分约束：余坯组板、余板组板中，支持化学成分的约束条件。每个订单包含各自的化学成分限制，每个余材（余坯、或余板）中包含化学成分的比例。当开启化学成分对比后，当对于一个余材来说，只有能其化学成分能够满足限制条件的订单，才可以使用该余材进行组板；

2) 钢种兼容约束：钢种改判是余坯组板、余板组板中常用的钢种兼容方式。有些时候希望钢种具有兼容性的方式来组板，有的时候不希望钢种改判。通过参数控制钢种改判是否生效；

3) 切头切尾量基准：支持通过板坯、大板、展宽等的信息判断切头尾量的大小；

4) 取样长度：订单支持取样长度，对于一个大板，取样长度取头、中、尾三个取样长度之和；

5) 切边量基准：通过板坯、订单、大板展宽等的相关信息判断切边量的大小；

6) 切缝量基准：通过大板信息判断切缝量的大小；

7) 组合长组合宽是否考虑切缝：组合长和组合宽，一般是将小板的长宽规格相加得到，但是有的时候，希望组合长和组合宽可以考虑中间的切缝的损耗，有的时候不希望考虑其中切缝的部分，可通过参数控制；

8) 厚度补偿余量：支持厚度补偿量配置表，在订单规格基础上，提供厚度补偿能力，辅助完成目标厚度达到规格要求；

9) 目标厚度考虑公差下限：在目标厚度的基础上可以配置公差上下界，可以满足目标厚度和公差下限配合，完成合理厚度计算；

10) 余板组板公差兼容：在余板利用时，支持目标厚度公差范围内和余板匹配的功能，进而完成公差范围内的厚度浮动组板；

11) 收得率下限约束：支持设置收得率下限，当大板收得率低于该值时，不作为组板结果输出。在计算过程中，该类大板会被拆散，并用于配合其他大板组合；

12) 坯长废料长度可控：支持余坯板坯长度废料限制约束，余坯组板中，废料长度上限通过参数配置；

13) 展宽比约束：可以通过展宽比参数，限制组板的合法性，当不在展宽比范围内时，组板不合法；

14) 连铸出钢材优先断面筛选：支持多优先级结晶器断面筛选，通过配置多种结晶器断面，完成优先使用第1结晶器尺寸组板组批；无法成功组板的合同，将采用第2结晶器尺寸、第3结晶器尺寸，依次类推。可支持多个结晶器断面；

15) 余坯余量均分：余坯组板中如果一块长坯，均分为多块短坯。这些短坯将均分剩余的坯长。但是在计算必要板坯长的时候，不考虑均分的这部分板坯长度，而使用真实板坯长。可通过开关控制是否生效。

优化目标方面，不仅仅是最主要的目标：收得率尽可能高；同时也会考虑到加工过程中的复杂程度，一般来说会考虑“相同或类似规格的订单的小板尽可能组在一张大板上”等，具体如下：

1) 相同合同（订单）优先在同一块大板中：相同的合同具有相同的钢种、相同的规格、物理化学属性要求，并且其中尺寸相同，也是剪切效率的重要影响因素。因此会鼓励相同订单优先组板在同一块大板中。通过参数配置，包括该功能是否启用、收得率下限门槛、额外的奖励分数、同合同数量的下限门槛。另外，同合同的分数，也可以采用计算公式的方式进行评估，会更适合某些用户的使用习惯：同合同分数公式中的 lambda ,分数=lambda * (alpha * 同合同小板数 - beta)，其中lambda、alpha及beta为可调节参数；

2) 相同尺寸小板优先在同一块大板中：同一块大板中的尺寸相同，是剪切效率的重要影响因素。有些订单虽然订单号不同、但是钢种和尺寸是相同的。因此会鼓励相同尺寸小板优先组板在同一块大板中。通过参数配置，包括该功能是否启用、收得率下限门槛、额外的奖励分数、同尺寸数量的下限门槛。另外，同尺寸的分数，也可以采用计算公式的方式进行评估，会更适合某些用户的使用习惯：同合同分数公式中的 lambda ,分数=lambda * (alpha * 同合同小板数 - beta) ，其中lambda、alpha及beta为可调节参数；

3) 倾向于一块大板中尽量放置更多的小板：一般来说，小板越多，加工过程约简约。虽然这不是必然的提升生产效率，但是从板坯、板的搬运等环节考虑，用户可能会希望将更多的小板集中在同一块大板中。通过参数配置，包括该功能是否启用、收得率下限门槛、额外的奖励分数；

4) 可以选择倾向更长的余坯：大多数情况下，余坯的利用长度越多越好，并且这个过程会保证长坯切短过程中，新的短坯的长度也是合法的。但是在一些特殊情况下，长坯中先使用的坯长越长，有时候会导致过短的新的余坯可能会难以找到合适的组板订单。总的来说，从流水订单池不断滚动的角度考虑，依然默认使用余坯的利用长度越长越好；

5) 余坯组板的大板收得率vs.板坯收得率：余坯组板场景，收得率指的是，大板的收得率。但是在优化过程中，使用的是板坯的利用率最大化。如果追求大板收得率最大化，需要通过参数配置；

6) S型组板中倾向于同宽的小板：S型组板时，可以选择倾向于同宽的小板。同宽的小板的优势是更适合产线生产效率。通过参数配置。

4.4.3 技术方案

自动组板算法：通过指定组板方式、轧制方式、成材率等核心目标，按照一定的规则，将不同规格的合同自由组合，组在同一轧制大板上，并输出相应的组板结果，支持功能如下业务流程中连铸出钢材组板、余材板坯组板、余材钢板组板三大算法功能模块：

1) 连铸出钢材组板：支持连铸出钢材组板，或者说炼钢组板。功能包括：坯长决策、轧制尺寸决策（组合长、宽）、订单选择决策；

2) 余坯组板：支持余材板坯组板，余坯转用充当。功能包括：轧制尺寸决策（组合长、宽）、小板（订单）选择决策；

3) 余板组板：支持余材钢板组板，余板转用充当。小板（订单）选择决策。支持全局最优解。

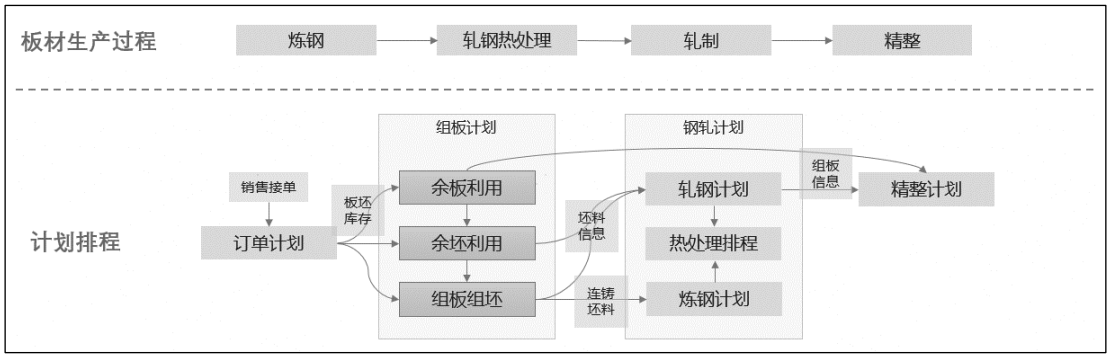

图4.11 钢铁板材组办流程示意图

多种组板模式：

1) A型（队列型）及细分类型：支持队列型组板，大板中包含的所有的小板会沿着纵向被排成一列。支持多种A型组板，功能包括：A1型组板：小板宽均相同。A2型组板：小板宽有差异；

2) S型（剖分型）及细分类型：支持剖分型组板，大板中包含的所有的小板会被拆成两列，大板中间会存在纵向的剖分线，两列小板会分别置于剖分线两侧。表示剖分剪。支持多种细分类型，S1：左右同宽，上下同宽，上下同长。S2：左右同宽，上下同长。S3：左右同宽。S6：仅维持中线剖分；

3) G型（火切型）：支持火切型组板，大板会被拆解为多个BLOCK块，BLOCK块之间沿着纵向排列，每个BLOCK块的宽度和大板宽度相同；大板中包含的小板被分配到BLOCK块中；在同一个BLOCK块中的1个或者多个小板平行摆放，且沿着横向拼接。该类型组板无法用剪机直接剪切，需要通过火切机裁剪。G是GAS的缩写；

4) 可定制其他类型：支持定制开发其他可配置的组板类型；

5) 组板类型的倾向性：在部分场景下，比如A型组板，由于考虑到生产效率，特备是剪切效率，常常希望牺牲一定的收得率，而得到更适合剪切效率的组板方式。一般来说，常用的优先级顺序是，A1>A2>S1>S2>S3>S6>G。通过参数配置完成组板类型和收得率之间的平衡。

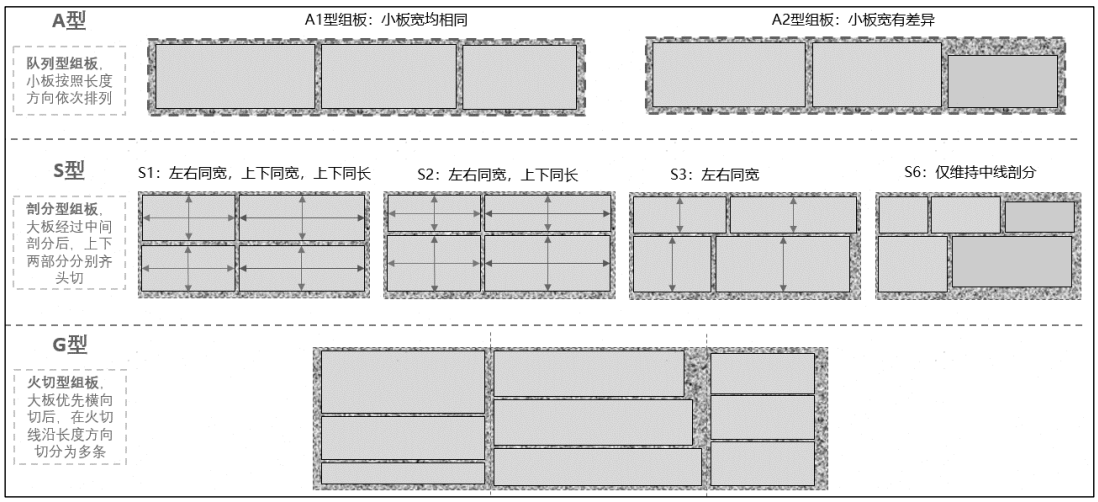

图4.12 组板类型示意图

多种轧制模式：

1) 纵轧轧制模式（L3）：支持纵轧轧制，支持板坯通过沿着长度方向进行纵轧。其中轧制方向为L，轧制代码为3；

2) 横轧轧制模式（C2）：支持横轧轧制，支持板坯通过一次90度旋转（转钢）完成横纵方向调换，然后在横方向上进行轧制。轧制方向为C，轧制代码为2；

3) 纵轧展宽轧制模式（L1）：支持纵轧展宽轧制模式，板坯进行两次方向旋转，实现纵向延长、横向展宽、纵向轧制三次轧制。大板纵向和板坯纵向保持一致。轧制方向为L，轧制代码为1；

4) 纵轧横轧模式（C4）：支持纵轧横轧轧制模式，板坯进行两次方向旋转，实现纵向延长、横向展宽两次轧制。大板纵向和板坯纵向相反。轧制方向为C，轧制代码为4；

5) 连铸板坯长度约束：连铸出钢材场景，支持板坯长度范围约束。组板的板坯长度，需要满足长度上下限范围约束；

6) 余坯组板板坯长度约束（长坯余料可用性保障）：余坯组板场景，支持板坯长度范围约束。对于长坯来说，当一个长坯坯自动切分为多个坯的时候，需要满足每一个短坯均在长度限制范围内；

7) 轧制最大长度：支持大板最大长度范围限制。组板过程中，大板长需要在最大长度范围内。生产中主要考虑的是大板移动过程中的场地范围限制。

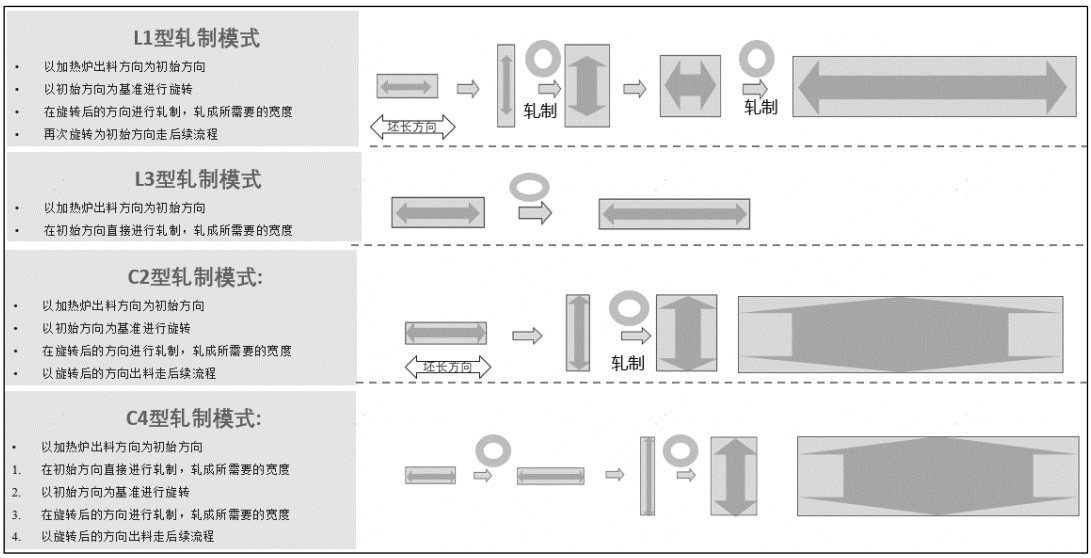

图4.13 轧制模式示意图

对于三种不同的场景，功能支持如下：（1）连铸出钢材组板，综合考虑全量可生产合同，按照规则需求组板，并输出相应的组板结果，支持上述所有组板模式、轧制模式；（2）余材板坯组板，先选定库存中的余材板坯，再匹配相应的合同，根据交货期选择订单进行组板消化，达到坯料消化的最优模式，支持上述所有组板模式、轧制模式；（3）余材钢板组板，先选定库存中的余材钢板，再匹配相应的合同，根据交货期选择订单进行组板消化，达到坯料消化的最优模式，支持上述所有组板模式。

集成模式：算法应用打包成镜像，包含连铸出钢材组板、余材板坯组板、余材钢板组板，部署在甲方提供的服务器。部署成功后生成RestFulAPI接口，相关接口可被应用系统（如MES）进行集成。

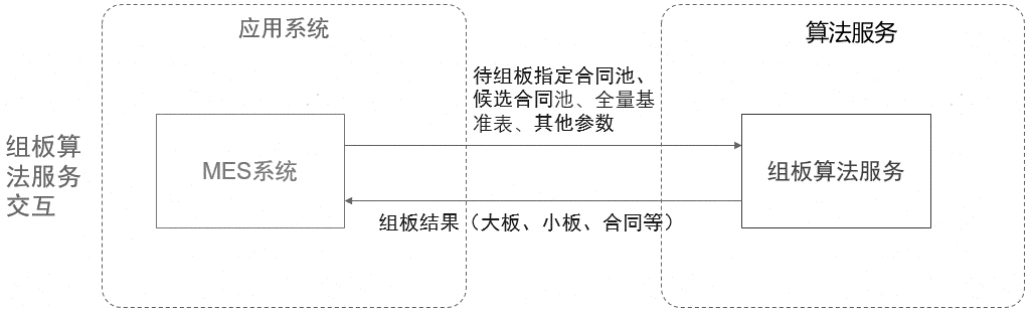

图4.14 集成模式示意图

4.4.4 应用成效

1) 多产线覆盖：支持宽厚板、中厚板；

2) 自动组板类型覆盖：支持多种组板类型自动组板（队列型\剪切型\火切型）；

3) 性能：千个合同分钟级组板（1分钟-10分钟）；

4) 收得率：连铸\余坯场景91%以上，余板单板全局最优解。

4.4.5 复制推广建议

1) 落地情况：2025年3月已在国内某TOP级知名钢铁厂上线，已在生产过程中使用；

2) 差异化竞争力：余坯组板。一般来说人工的余坯组板收得率为84%，该产品设计之初收得率目标为87%，最终落地收得率为91%，得到客户好评。在余坯组板场景，该效果为同行业高水准，是该产品和友商竞争中的关键护城河；

3) 成熟度高：已覆盖大部分的业务需求；支持新功能需求定制；

4) 支持快速POC：仅需客户参照“POC测试接口表”提供测试数据即可；已有较成熟的技术协议模板。

4.5 供应链零售门店采购建议优化技术方案与应用实践

4.5.1 业务背景介绍

门店订购，主要服务于便利门店进行智能销量预测和备货指导，通过AI能力辅助门店店长和业务员更快速、更精准经营门店业务。为了给第二天的商品销售做充足的库存准备，当前门店每天都需要依靠人工经验来制定订购策略。为了减少人工订货预测水平差异带来的缺货或积压现象，降低对大量SKU订货的人力需求，本专案期望使用预测大模型+天筹求解器的组合方案来实现门店订购，结合门店库存、在途、货架、历史销量等信息给出最佳订货量建议，达到缺货和库存成本的最优解。

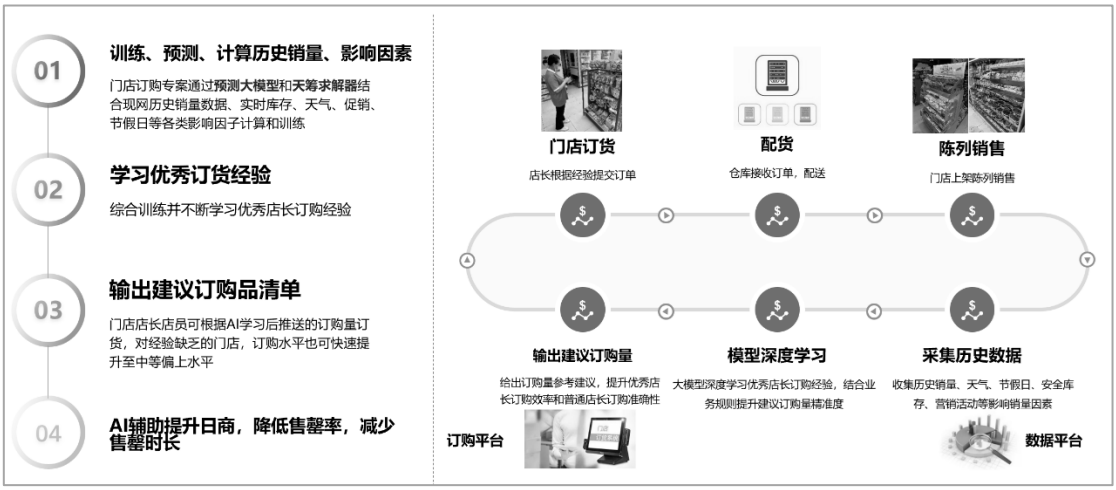

图4.15 智能订购流程示意图

4.5.2 业务需求

业务现状：（1）具备信息化：终端设备为手持设备，库存可视化，在途信息可视化，折损可视化，历史订购：周月级历史订购记录；（2）规则辅助订购：后台具备规则引擎；可参考历史订购同期；机会损失有欠品时间带提示，有折损量提示；（3）人工经验确认:参考建议订购量，店长参考实体货架陈列量，在建议订购量基础上修改，每天多次订购，以最终次为准。

业务痛点:（1）业务量大: 门店数量：全国3000+门店；鲜食量级：每店100+种活跃鲜食非鲜量级：每店1000+种非鲜食；订货频率：店长日级订购；送货频率：当天订购、T+1送货、T+2销售；（2）智能化程度低: 规则引擎来源：来自店长访谈；区域依赖：对访谈区域有依赖；时效性：无法根据销售趋势调整；地域性：无法根据区域差异调整；总结：规则配置响应慢；（3）依赖人工经验: 建议量策略更新不及时；每天人工修改工作量大；建议订购量不准确，需要人工介入的较多，依赖人工经验。。

业务核心诉求:（1）差异性订购: 精准订购：鲜食匹配店长意愿；优秀经验：非鲜食学习优秀店长；实时推荐：小时级推荐；（2）高时效推荐: 时效性：满足推荐贴合近期趋势；地域性：满足各个区域的特点；（3）减少干预: 精度：建议准确；修改：减少人工修改数量。

4.5.3 技术方案

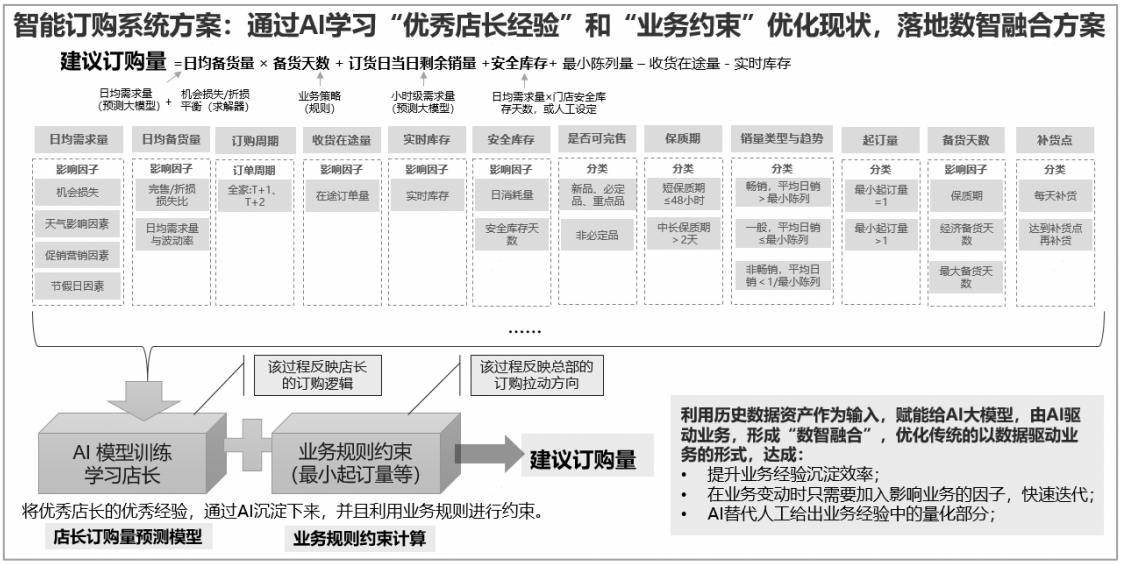

图4.16 智能订购解决方案

该解决方案是一个数据资源和智能化技术融合的过程，由传统业务架构的数据驱动业务，转变成 由数据给AI、大模型、求解器做输入、赋能，再借助AI强大的计算能力驱动业务，更加精准的计算、推理，形成业务的全链路打通，避免数据孤岛。在门店订购专案中我们通过两个步骤来处理AI和实际业务的关系。

第一步，选择优秀店长作为学习的样本，并且将业务上考虑到的因子，特别是店长访谈中提及的因子，都加入到模型中用AI进行学习。

第二步，但是AI给出的结果，未必总是符合预期的，最明显的就是，有时候，我们的最小订货单元是一个确定的量，比如说啤酒，有些品牌的啤酒，总是6个一包进行订货。那么当AI告诉店长需要订货为11个的时候，我们需要根据业务经验来判断，实际上AI是希望我们订购2包12个。也就是说，需要利用业务规则，对AI的计算结果进行修正。

这样除了省去一些工作量，还有其他的优势：（1）提升业务经验沉淀效率；（2）在业务变动时只需要加入影响业务的因子，快速迭代；（3）AI替代人工给出业务经验中的量化部分。总的来说，训练AI大部分的工作是在后端完成的，是处理数据，充分利用了历史数据的宝贵资源。另一方面，AI+规则，也是将传统业务+新技术相结合的一种尝试。

其中学习优秀店长经验的过程中，模型特征采用: 当前库存因子、当日消耗量

因子、订销比、后天是否参与活动、基本订购单位因子、今天是否参与活动、近期的周级别的订销比、门店id、门店id因子、明天是否参与活动、趋势因子、日均消耗因子（来自大模型的预测结果）、日期因子、商圈类型、鲜食因子、销售类型因子、效期因子、在途库存因子、在途时间因子、最大陈列量因子、最适陈列量因子、最小陈列量因子、最小起订量因子、昨日销售量和上周同日销售量环比。

模型预测店长订购量之后，通过如下规则约束，对推荐结果进行修正：（1）通过历史7天的最高单日销量，修正鲜食建议订购量；（2）对于新品，利用同类型超A类型的近期日均向量，修正鲜食、非鲜食的建议订购量；（3）考虑基本订购单元大小和基本订购倍数等，修正鲜食、非鲜食的建议订购量；（4）考虑在售商品的最小陈列量，修正鲜食、非鲜食的建议订购量。

通过上述AI推理模拟店长行为，再利用上述业务规则约束，最终给出规则修正后的建议订购量。

4.5.4 应用成效

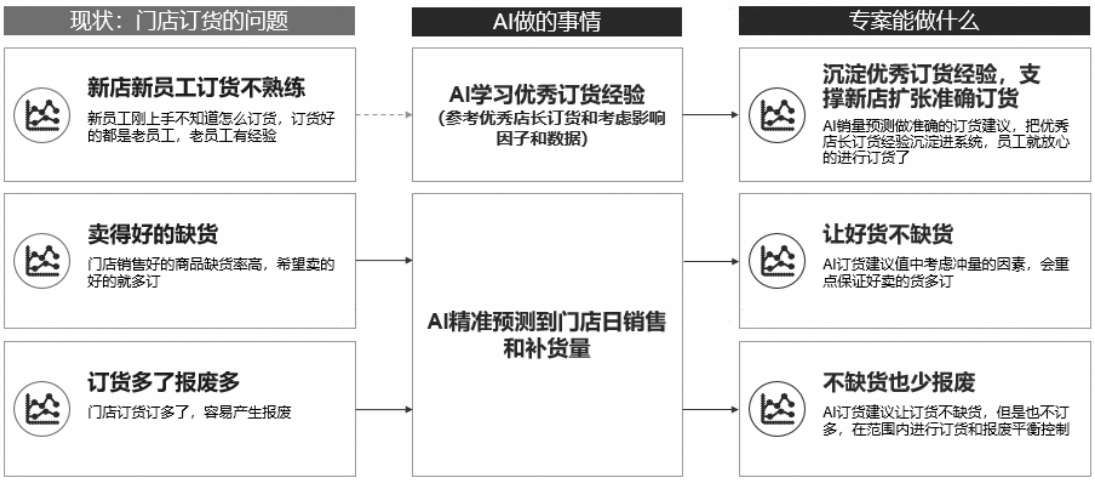

图4.17 智能订购效果示意图

实现如下能力：（1）差异性订购：鲜食匹配店长意愿；非鲜食学习优秀店长经验；小时级推荐；（2）高时效推荐：满足推荐贴合近期趋势；满足各个区域的特点；（3）减少干预：建议准确：减少人工修改数量；其中非鲜食准确率（偏差≤3）90%以上；鲜食（偏差≤2）82%以上。

4.5.5 复制推广建议

主要的推广策略包括强调以下三方面：（1）便利店场景为主，鲜食场景为主，主要的原因是，鲜食产品为快消品，保质期较短不能长期保存，店长的决策空间较小。相反的是非鲜食，会导致囤货对收益的影响不大，货物不容易过期；（2）在之前的案例中，输入数据包含的种类很多，不一定所有的都要用，当前案例使用的特征已经过实践检验，可以作为POC的参考；（3）本算法中考虑了“报童模型”的变种算法，综合考虑了错失销售机会和鲜食过期报废的影响，结合了预测、决策两种算法技术，此类AI算法的综合应用有可能成为该场景的亮点。

4.6 铝加工冷轧排程智能调度技术方案与应用实践

4.6.1 业务背景介绍

4.6.1.1 业务场景描述

在铝卷加工行业，冷轧加工做为完成成品的最后一个环节，包括冷轧线、退火线、精整线3条产线[145]。冷轧线主要处理开坯冷轧、中间冷轧、成品冷轧工序，退火线主要处理中间退火、氮气保护退火、连续退火等工序，精整线主要处理清洗、矫直、切边等工序。不同产品的工艺路线可能不同，但是其基本的生产流程基本一致，其相关工序流程如图4.18所示。其中，不同产线作用描述如下：

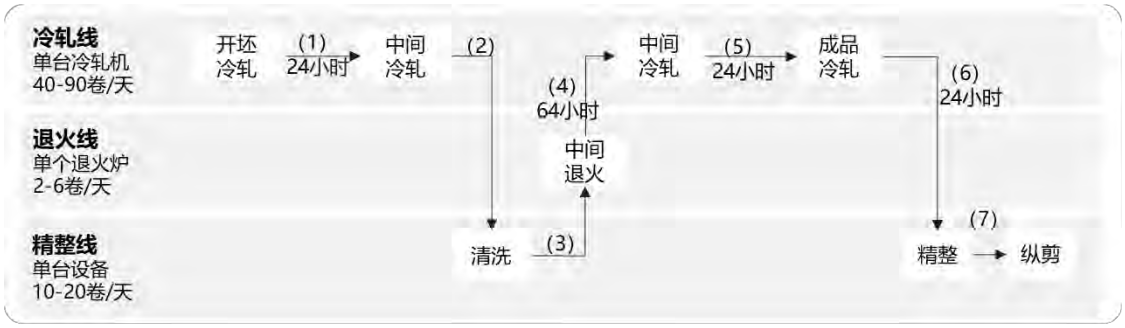

图4.18 工序生产流程示意图

1) 冷轧线：将准备好的铝板坯料送入冷轧机，冷轧机的两个轧辊对铝板施加压力，使得铝板的厚度逐渐减薄，同时由于加工硬化，铝板的强度增加、塑性降低；

2) 退火线：根据最终产品对铝板厚度和性能的要求，可能需要进行多次冷轧

和退火的交替操作。每次冷轧都会使铝板更薄、强度更高，退火则恢复铝板的塑性，防止铝板在轧制过程中破裂；

3) 精整线：冷轧后的铝卷经过裁剪、清洗、矫直等步骤，按照客户要求的尺寸进行切割。然后进行检验，检查铝卷的厚度、表面质量、力学性能等指标是否符合产品标准。合格的产品进行包装、标识，等待发货。

由于冷轧环节涉及产线工序复杂，生产跨时较长，人工排产计划难以应对，因此需要引入智能化排产工具以应对生产多品种、小批量、多工序的业务特点。

4.6.1.2 业务痛点

当前铝加工冷轧排程场景存在如下痛点：

1) 业务体量大，单次排产计划涉及800-2000卷，7000-15000道工序，50+机器，生产跨2个厂区，人工排产比较困难；

2) 人工排产依赖专家经验和简单的逻辑计算，通常需要进行多次调整，且难以兼顾组辊、组刀等40+复杂切换规则；

3) 人工排产对组炉考虑不合理，存在10%左右组炉计划不可执行；

4) 面对插单情况，人工难以快速调整排产计划；

5) 人工排产难以综合考虑产能、交期满足率、冷轧换辊等目标。

4.6.1.3 技术挑战

首先，不同产线存在复杂的生产规则要求，

1) 冷轧线：

a) 轧辊轧制铝卷顺序要求由宽到轧，否则需要换辊；

b) 不同轧制粗糙度的铝卷不能放在同批次轧制；

c) 不同工序不满足轧制顺序需要换辊；

d) 轧辊超过使用次数上限需要换辊；

e) 连轧工艺（连续处理的冷轧道次）需要在同一机器上加工。

2) 退火线：

a) 退火炉一次可以同时处理多个工序相同的卷，处理时间以同批次卷所需最长处理时间为准；

b) 退火时间通常需要1-3天，组炉利用不合理通常会造成加工堵塞，影响整体排产周期。

3) 精整线：切换规则与产品属性、机器类别有关，涉及多种复杂切换规则。

其次，产品生产本身存在可替换工艺路径，涉及跨厂区生产，如何通过合理的工艺生产路径选择，加工机器选择对平衡产能、交货期、切换和转厂目标也是一项重要挑战。

4.6.2 业务需求

4.6.2.1 客户整体功能需求

面对上述业务痛点，客户提出如下业务核心诉求：

1) 排产合理性优化：排产计划符合生产规则约束，退火组炉合理且可执行，转厂生产次数和冷轧换辊次数合理符合业务目标；

2) 多目标优化：实现最大化产能、最大化交货期满足率、最小化切换次数目标；

3) 排产效率优化：提升排产计划可执行率，减少人工调整干预，满足临时插单需求。

4.6.2.2 集合和常量

铝加工冷轧排程场景涉及的关键数据集合和常量如下：

1) 铝卷信息：产品类型、合金类型、已转运次数、订单优先级、前置工序所在机台、前置工序完成时间、交货期时间、卷径、重量、长度、工序信息等铝卷基础数据；

2) 设备信息：设备维修时段、设备产能数据、设备与工序可达关系等数据；

3) 工序信息：工序可用设备、工序间隔数据、连轧时间间隔限制数据、工序设备白名单和黑名单数据；

4) 规则信息：切换规则信息、辊使用限制数据、退火炉组炉规则数据、冷轧切换规则数据等规则数据。

4.6.2.3 业务目标

表4.6为铝加工冷轧排程场景的业务目标概述。

表4.6 冷轧排程场景业务目标

目标优先级	目标	描述
1	产能	优先考虑最大化产能
2	交期满足率	最大化交期满足率，尽可能满足交期
3	冷轧换辊次数	冷轧切换次数尽量少
4	过渡板使用次数	过渡板使用次数尽量少
5	满足机列优先级偏好	每个工序对可用机台有优先级偏好
6	跨厂转运次数	能不跨厂就不跨厂
7	退火组批数	进行合理组炉且组炉批数尽量少
8	退火组机台利用率	退火炉闲置少
9	精整换刀次数	精整换刀次数尽量少

4.6.2.4 业务约束

表4.7为该场景具体生产约束。

表4.7 冷轧排程场景业务约束

序号	业务规则约束&功能要求	描述
1	工序-机列可达约束	工序可用一组机列满足可用约束
2	产品-机列可达约束	产品可用机台满足机台支持的卷径范围、产品型号等物理属性
3	产品-工序-机列可达约束	针对产品的某工序对可用机台有限制
4	卷最早达到时间约束	每个卷存在最早可加工时间
5	机台设备地点及转运	当卷生产涉及的设备在不同厂区时，涉及转运时间，该转运时间与时间间隔约束不重叠
6	工艺路径与替代	每个产品有自己的工艺路径，部分产品部分工艺路径可被其它工艺路径代替
7	机台不可用时间	设备检修时不能进行排产
8	机台产能利用率	机台每天可用时间满足产能利用率，其不可用时间需处理为一段连续的不可用时间区间
9	退火炉容量限制	不同型号退火炉支持放入的卷有尺寸要求，且单批次对放入的卷有数量限制
10	工序间隔时间	同卷部分工序间加工存在最小和最大时间间隔要求
11	工序生产时间	工序生产时间满足机台生产能力
12	连续退火工序切换约束	连续退火工序需满足生产厚度差、宽度差、温度差规则，否则需要进行切换
13	矫直工序切换约束	矫直工序生产需满足是否矫直切换规则
14	套筒切换约束	卷套筒生产需满足是否套筒切换规则

15	涂油切换约束	卷涂油生产需满足涂油类型切换规则
16	冷轧切换约束	冷轧工序生产需满足宽度切换、工序切换、粗糙度切换等切换规则
17	精整切换约束	精整工序生产需满足精整切换规则
18	连轧约束	卷连续冷轧工序需满足连轧约束，即在同一机台上加工
19	任务所在工序	卷同时刻只有一个工序在加工
20	任务优先级	卷生产需满足任务优先级，比如插单、高价值产品需优先生产
21	排程周期展望	主要考虑给定排产周期内计划最优
22	锁定排产	部分计划工序还未执行，但已在准备阶段无法调整，需要锁定在指定机器上
23	大卷拆小卷	卷因卷径和重量原因无机台可用时，尝试拆卷生产；人工可指定拆卷产品类型及工序，可配置
24	机器建议检修时间	根据算法排产结果返回可用于检修的空闲时间区间，需满足参数配置
25	转卷限制	转厂时满足同一卷转运不超过2次的限制；需考虑做计划时卷已转运情况

4.6.3 技术方案

4.6.3.1 算法服务方案设计

如图4.19所示，通过将ERP、MES中的数据整合汇总提供给求解器服务引擎，再将求解器服务引擎计算的排产计划结果展示到前端界面，由人工调整确认后下发到MES中进行计划执行，形成端到端的智能化决策体系。

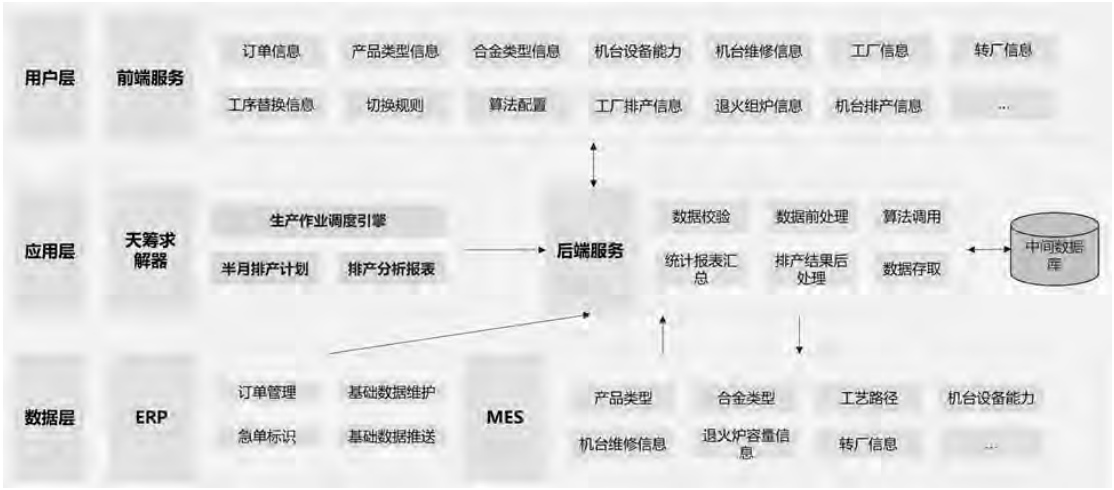

图4.19 应用架构及调用关系示意图

4.6.3.2 算法设计

基于对业务规则的梳理，将主要业务规则转换为具体的生产调度问题特性[146]，如表4.8描述：

表4.8 业务规则与问题特性对应关系说明

问题特性	业务规则
柔性作业	工序—机列约束关系
柔性路线	工艺路径与替代
并行组批	退火炉容量限制
切换时间	XX切换约束、机台地点及转运
最小时间间隔	工序间隔时间
最大时间间隔	工序间隔时间
释放时间	卷最早达到时间约束
固定不可用时间段	机台不可用时间、机台产能利用率
OutTree工艺路径	大卷拆小卷
任务优先级	任务优先级、锁定排产
最大产能目标	最大化产能目标
拖期目标	交货期目标
切换次数目标	冷轧换辊、过渡板使用次数

如图4.20所示，根据输入数据以及对问题特性的梳理，构建合适的数学模型，基于启发式、元启发式、混合整数规划等方法，进行求解，给出具体的排产结果，具体实现路径如下：

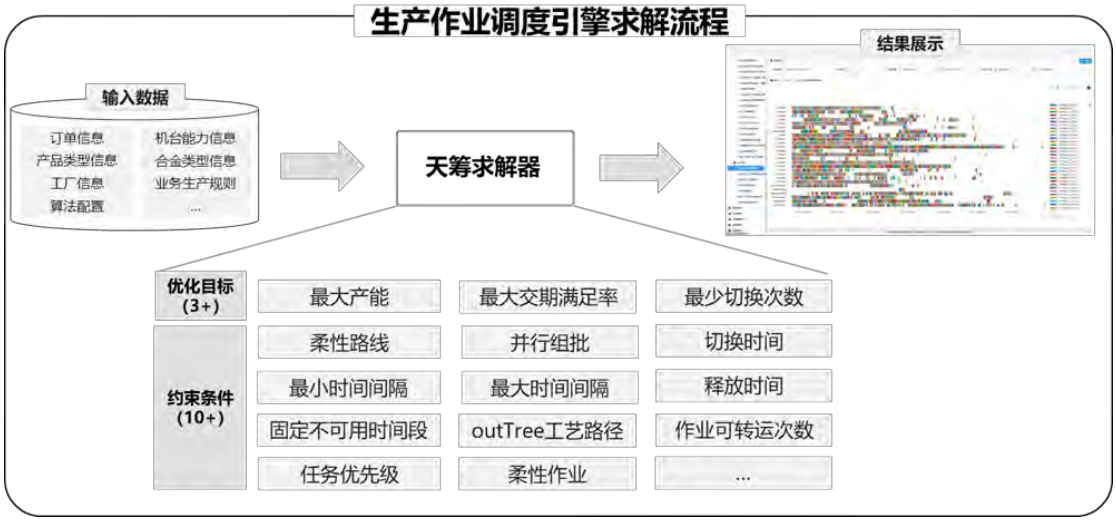

图4.20 生产作业调度引擎求解流程示意图

1) 模型构建：根据订单信息、工序信息、机器信息等基础数据，结合业务规则，

建立满足上述问题特性的数学模型，并以“提高产能、减少拖期时间、减少切换次数”为核心目标，构建目标函数；

2) 模型求解：借助天筹AI求解器生产调度引擎等专业求解工具，对构建的数学模型进行高效求解，确保高效求解以及求解结果的最优性；

3) 方案输出：根据求解结果，生成符合实际业务需求的排产计划方案，明确各产线机器的排产计划、各订单的具体加工时间、铝卷转运计划、退火机器的组炉情况、冷轧机器的组批情况等关键排产信息。

4.6.4 应用成效

引入天筹AI求解器后，冷轧排产由人工月、日计划升级为模型自动排产，并达到如下效果：

1) 分钟级求解能力：单次排产时间由人工天级降至算法分钟级，为计划员制定和调整排产计划提供充分时间；

2) 提升业务效果：冷轧每辊平均道次数由18道次/辊提升到20.7道次/辊；产能提升约1%-2%；计划合理性与精度显著提升；

3) 泛化性高：业务规则可配置，支持跨厂生产等多种场景。

4.6.5 复制推广建议

该排程方案主要适配铝加工行业冷轧生产排程场景，因此主要考虑在铝加工行业进行复制推广，对于该场景在不同铝加工厂商不同业务目标需求，需进一步评估场景特性的通用性以进行功能适配，或通过额外后续方案解决。

对于其它行业，如钢铁等工序流程相似的轧制生产排程，该方案具有一定复制的可能，具体场景仍需进一步评估。

4.7 烟草行业卷包计划与制丝排程技术方案与应用实践

4.7.1 业务背景介绍

4.7.1.1 业务场景描述

烟草加工是一个融合农业与工业的复杂过程，通常以烟叶为原料进入卷烟厂加工。烟叶会根据产品设计进行配方组合，并通过制丝工艺（包括润叶、切丝、烘丝、加香等）加工成合格的烟丝。最后，烟丝通过高速卷接设备制成烟支，并完成包装，成为上市销售的卷烟产品。

当前烟草行业正经历从传统制造向数据驱动的智能化生产模式转型，这一系统性变革由市场环境变化、技术革新、政策引导和企业内生需求四重动力共同推动。同时烟草行业对数字化转型需求迫切，烟草行业面临全球化竞争加剧和消费者需求多元化的双重挑战，传统生产模式在效率、成本和合规性方面已显现明显局限。烟草生产具有工艺复杂、标准严苛的特点，构建数据驱动的智能生产体系已成为行业发展的必然选择。

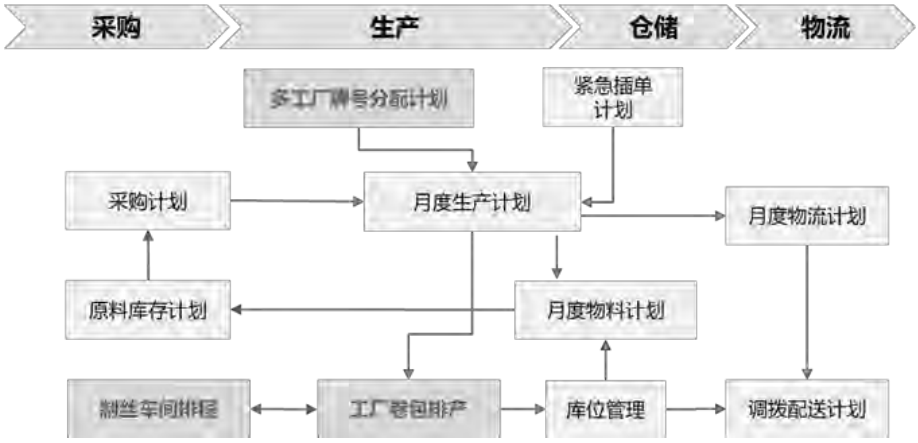

图4.21 烟草计划业务流程示意图

烟草厂主要分为卷烟和制丝两个环节（见图4.21）。卷烟草通常的计划流程中，首先由省公司下达月生产计划，工厂接收并分解为日生产计划。接着，生产处制定制丝批次计划，包括拆包、叶丝和丝线计划。制丝车间根据计划执行工艺段、香精香料和梗线计划。同时，卷包车间制定卷包生产计划和换牌计划，确保生产顺利进

行。整个流程环环相扣，确保生产高效有序。

烟草生产依赖稳定的原料供应链（如烟叶种植基地）与多环节协同（制丝、卷包、物流），但传统模式下，省公司、工厂、车间之间的信息孤岛导致资源分配效率低下，难以实现跨区域产能共享与动态调度。

4.7.1.2 业务痛点

烟草行业计划和排产的主要业务痛点总结如下：

1) 手工排产耗时，需要反复调整，依赖专家经验，难度大，耗时长，沟通成本高；不同专家排产习惯不一样，每次排产的习惯差异可能导致计划一致性较差。

2) 动态调整与异常响应滞后：原料供应延迟、设备故障、订单临时调整等不确定性因素频发，但传统依赖人工经验的排产系统缺乏快速调整能力；如省公司需求变化或紧急查单，难实现计划快速重排响应。

3) 贮丝柜储存时间安排难以决策，缺少生产进度匹配以防止出现停机情况。

4) 换牌换线频繁导致效率损失：卷烟品牌多样化要求频繁切换生产线，但换牌操作涉及设备清洗、参数调整等耗时环节，未充分优化换牌顺序，导致设备闲置或工时浪费。

4.7.1.3 技术挑战

1) 卷包计划需满足不同场景下可配置化和求解能力，对多工厂分配、安全库存水位、开台计划进行综合成本优化。在分工厂计划中需要更多考虑对省计划的满足行，以及具体资源（班次、人力、辅助设备）的资源分配优化。

2) 制丝排程（见图4.22）从模型的三要素变量、约束、目标角度看均有一定复杂性。变量角度：需要决策工艺路线选择与工序开始结束时间，决策空间大。约束角度：no-wait约束、Q-time约束以及工序分流合并约束使问题的约束满足异常复杂。目标角度：需要在不断丝的基础要求上，机器生产具有连续性、均衡性，且尽量满足牌号优先级等额外高阶目标，这类目标的建模求解均有相当的复杂度，对性能和最终效果影响很大。现有文献和技术往往针对某单一特定场景[147][148]，如何通过合理的工艺路线选择、存储/发酵时间安排、作业顺序和生产节奏来满足工艺发酵要求并平衡各个目标是制丝场景的核心挑战。

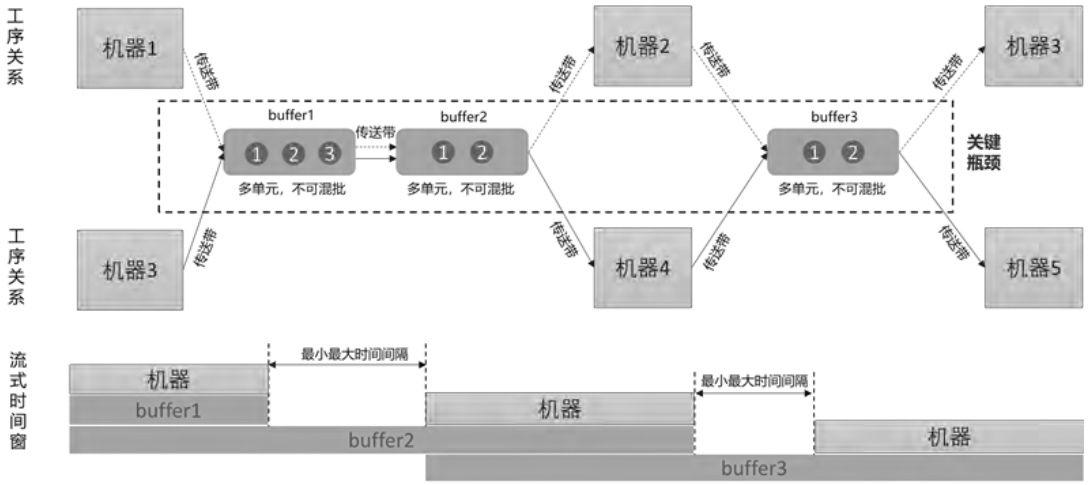

图4.22 制丝场景示意图

4.7.2 业务需求

4.7.2.1 业务场景目标

烟草计划和排产的主要业务场景目标总结如下：

1) 智能调度代替人工,降低人工工作量，生产计划实现自动化，减少人工排产工作量；

2) 通过AI算法优化公司计划和工厂生产排程,实现公司和工厂的生产效率最高；

3) 优化生产调度,利用AI算法,有效减少生产中的闲置时间，到“生产计划编程实现自动化”和效率最优。尽可能少的换牌和开台损失，保证生产稳定性和连续性；

4) 卷包计划需输出每个牌号的每日计划量以及使用资源关系；制丝排程需给出精确的工序开始与结束时间，并决策使用的工艺以及柜体的贮存时间。

4.7.2.2 集合和常量

卷包计划场景涉及的关键数据集合和常量如下：

1) 产品信息：每个产品的需求、库存、安全库存要求以及可生产机器和效率等基础产品信息或业务数据；

2) 机器信息：包含机器编码、机器数量和机器初始状态等信息，以及每个机器在不同时间的最大产能；

3) 辅助资源信息：包含喂丝机、班组数量限制等，以及辅助资源与产品、机器

的关系；

4) 成本信息：包含优化目标中的成本参数，包含库存成本、换型成本、开机成本、延期成本、环境能耗等成本参数；

5) 换型矩阵：机器中不同产品生产的换型时间矩阵。

制丝场景涉及的关键数据集合和常量如下：

1) 烟丝批次信息：每批烟丝编码、重量、优先级、所属牌号、可用喂丝机、喂丝速率及期望的喂丝时间；

2) 工艺路线信息：每个牌号可选的工艺路线，每条工艺路线的各个工序信息，每道工序需要占用的机器资源、柜子资源、传送带资源等其他辅助设备资源、机器的加工速率、柜子的可用编号、前驱工序和后继工序，工序间的最小最大时间窗要求；

3) 机器信息：机器编码，机器数量，机器每加工一批后需要的清扫/维护时间，是否受班次时间约束；

4) 柜子信息：柜子编码，柜子数量，柜子容量，是否对顶柜，是否受班次时间约束；

5) 辅助资源信息：辅助资源编码（主要是传送带），辅助资源数量；

6) 班次信息：班次时间。

4.7.2.3 卷包计划场景功能

对于卷包加工的计划场景的主要业务约束在表4.9中列举。

表4.9 卷包加工场景业务约束

序号	功能名称	详细描述
1	码段约束	本月的月度生产计划总产量应等于下发的码段数
2	调拨量约束	各规格/牌号交付量应满足月度调拨量需求
3	交付时间要求	各规格/牌号需要在交付时间要求内交货
4	产量进度要求	每个工厂当月的产量进度应满足省公司下发的进度要求
5	指定规格/牌号的计划生产量应满足批量要求	1、指定牌号满足批量整数倍生产，有些牌号同一批量在不同工厂之间可能存在差异。
6	根据原材料的到货时间制定相应的生产计划	根据不同牌号对应原材料的到货时间，生成卷包计划，避免因为原辅料缺失导致生产计划的执行受到影响。
7	工厂最大开台面限制	即各工厂每日不能超过本日的最大开台面数量限制。
8	设备约束	1、设备可以生产的卷烟牌号及生产效率不同，单台机器

		同时只能生产同一个牌号。 2、设备生产不同规格/牌号之间换型时需要时间消耗。 3、尽量用停产、轮保、假期结束的第一天进行换牌
9	喂丝机和卷包机连接关系约束	1、喂丝机和卷包机之间是一对多的关系；2、一台喂丝机最多可以同时连接x台卷包机；3、一台喂丝机同时只能给生产同一个牌号的卷包机喂丝。
10	工作日历约束	要求在工作日历时间内完成生产。
11	维修维保日历约束	生产计划避开维修维保时间段。
12	安全库存要求	1、各牌号/规格应尽量靠近给定的安全库存值 2、安全库存为一个范围值，超出一定数值后有阶梯惩罚
13	月度生产计划约束	各工厂需生产的牌号及数量需精准满足月度生产计划
14	机台开启消耗	每台设备都能够单独设置开机时间损耗和成本损耗

4.7.2.4 制丝排程场景功能

对于制丝排程计划场景的主要功能需求在表4.10中列出。

表4.10 制丝排程场景业务约束

序号	**功能名称**	**详细描述**
1	工艺路线选择	每个作业可能定义多条可选的工艺路线，系统需自动为每个作业选择且仅选择一条最优路线进行排程。
2	工序分流汇聚	支持工序的入树和出树结构。
3	衔接约束	根据前后工序使用的资源类型（机器M或缓冲B），必须满足特定的衔接规则：1. M→M: 齐开齐停。2. M→B: 齐开晚停。3. B→M: 晚开齐停。4. B→B: 紧密衔接。
4	Q-Time	任意两道工序之间需满足特定的时间间隔限制 。
5	辅助资源容量	工序不仅要占用主要资源，还要占用辅助资源。
6	缓冲资源容量	缓冲类资源允许多个任务共享，但受总容量限制。对于汇聚节点，需精确计算不同分支到达后对容量的分段占用，且不同任务不可混装。
7	不可用时间	资源存在预设的不可用时间窗。
8	准时交付	作业的最后一道工序必须严格按照喂丝时间开始，保证不断丝。
9	优先级	在同一天内，作业应该按照优先级执行。
10	每日连续生产	机器在每天的开工时间尽可能连续，避免反复启停
11	每日负载均衡	控制同一资源在每一天的任务数量波动。
12	存储时间冗余	在满足硬性 Q-Time 约束的前提下，使得实际存储时间尽可能接近理想值（通常为最大最小值的中间），而非卡在边界值。
13	滚动排程	排程输入考虑柜的状态、发酵状态、机器输出状态。
14	班次时间	机器只能在班次时间内生产

4.7.3 技术方案

4.7.3.1 技术架构设计

为解决烟草行业排产难点，通过构建烟草高级计划与排程（APS）技术架构（见图4.23）针对不同环境问题提供对应技术放哪。整体结构上通过上层应用、天筹AI算法引擎、数据中台和算力底座四个主要部分实现技术方案搭建。其中APS应用包括需求计划、省公司计划、卷烟厂计划、车间排程、物料管理和仓储物流管理，确保生产流程的高效协调。天筹AI算法引擎提供需求计划、生产计划、生产排程、运输计划和需求计划的引擎配置，优化生产决策。

基于客户痛点场景，宜重点使用多工厂卷包和制丝排程算法引擎为核心的智能排产解决方案，通过行业模板和算法引擎配置化，形成可复制技术方案。

图4.23 烟草高级计划与排程技术架构

4.7.3.2 算法设计

卷包省计划分配模型通过构建包含多目标（如换型成本、生产成本、库存成本等）和多约束（如产能约束、模具使用、生产换型等）的MILP模型，实现对生产计划的精确优化。该模型能够处理复杂的业务规则和约束条件，确保在满足省公司计划和分工厂需求的同时，实现资源的最优分配和成本的最小化。

1) 工厂卷包排产（见图4.24）采用启发式算法与天筹求解器结合进行求解，合理分配机器资源和辅助资源分配，尽可能降低整体班次使用，以及齐开齐停分配，

提升计划的灵活性和响应速度，有效应对动态调整和异常响应的挑战；

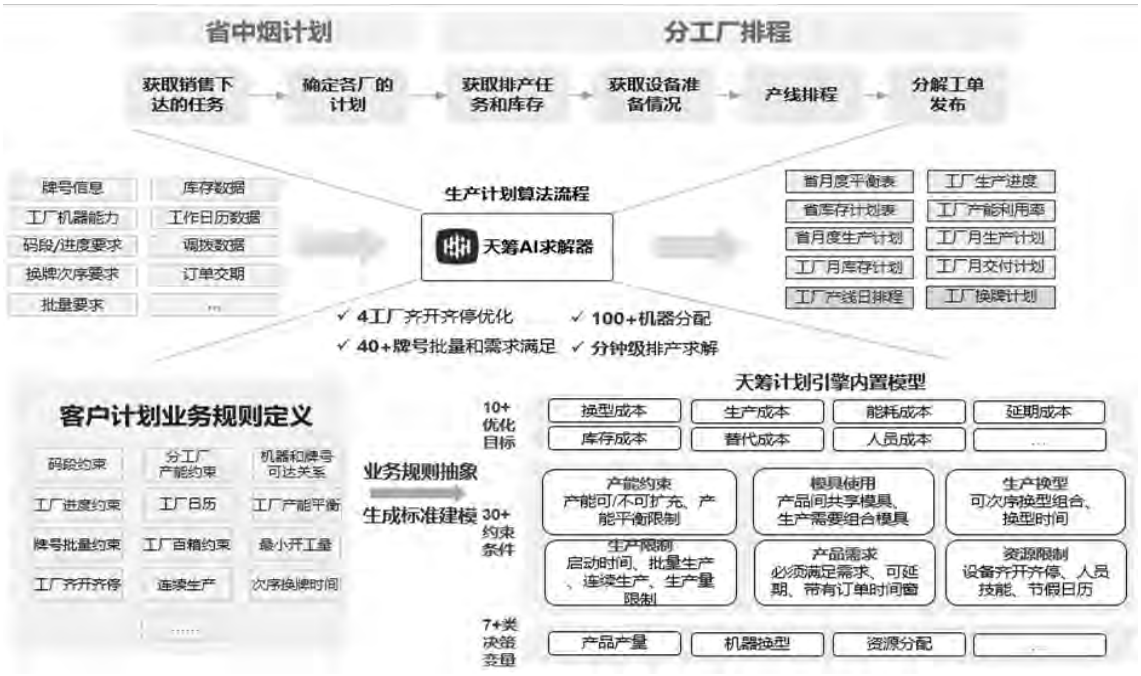

图4.24 卷包计划算法设计示意图

2) 制丝排程（见图4.25）通过“启发式+约束规划天筹求解器+后处理”的混合算法实现排程问题的快速求解，启发式负责任务调度以及时间预剪枝，天筹约束规划求解器负责时间推演，后处理负责对解进行进一步优化，实现多目标的优化权衡。

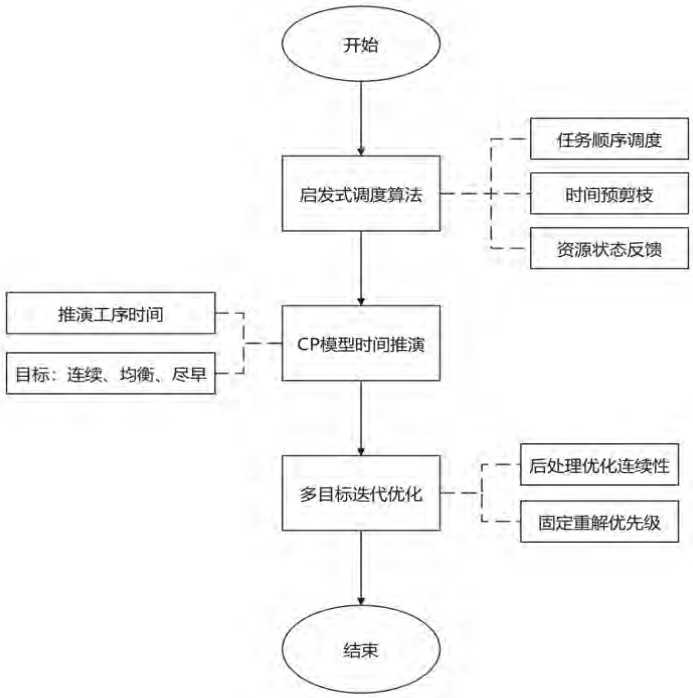

图4.25 制丝排程算法流程示意图

4.7.4 应用成效

多约束条件下的最优生产计划：

1) 智能排产平均每个月可以减少1天的生产时间，减少1.5次换牌；

2) 工厂齐开齐停率平均可达80%以上，预计节省能源费用+节省换牌损耗上百万每年；

3) 满足智能工厂建设要求；

4) 智能排产方案包括公司月计划、工厂卷包计划、工厂制丝排程、计划量化评比、卷包滚动排产、制丝滚动排产等13个子场景。完全满足智能工厂有关排产的建设要求。

分钟级级结果输出能力：

1) 卷烟厂排产缩短为10~20分钟自动化求解省工厂和分工厂计划；

2) 制丝排程实现秒分钟求解，相比人工排产效率极大提升，排程展望周期更长；

3) 排程连续性、均衡性和优先级违反均实现优化，部分产线实现完全连续。

4.7.5 复制推广建议

烟草方案宜在不同省份的卷烟厂进行推广，如跨行业适配性较弱。其中卷包计划和制丝排产可可通过行业模板在不同卷烟在进行适配，如存在不同卷烟厂的特殊排产限制，需进一步评估是否可以作为通用行业特性功能开发，或搭配外层技术方案满足。

在其他行业领域中，需依据是否具有相近流程，尤其是贮丝柜、储叶柜相似的贮存工序的加工场景。例如食品发酵加工行业需要进行发酵处理，制药生产中有陈化、醇化或稳定化处理的工序，具有一定复制本文制丝排程算法的相似条件。

4.8 空调钣金加工计划与排程技术方案与应用实践

4.8.1 业务背景介绍

4.8.1.1 业务场景描述

在家电行业中，金属板材的成型加工是生产流程的核心环节。钣金成型加工是将金属板材（如不锈钢、镀锌板等）通过冲床、折弯机等设备加工成特定形状的零件或产品。例如冰箱的外壳、洗衣机的底座、空调的外壳框架等，通过成型加工实现家电的外观设计需求，并提供整体结构强度，保护内部组件。

如图4.26所示，成型加工根据设备能力和钣金类型，分别在多条产线上完成，每条产线由多台冲床串联组成（视为整体单元）。钣金成型加工中每种零件需要专用模具，且一般在同一班次只能在一条产线生产（避免模具重复使用冲突）。如需要切换生产的钣金类型，需要进行换型操作，在切换零件原料的同时更换模具，通常会消耗时间和成本。

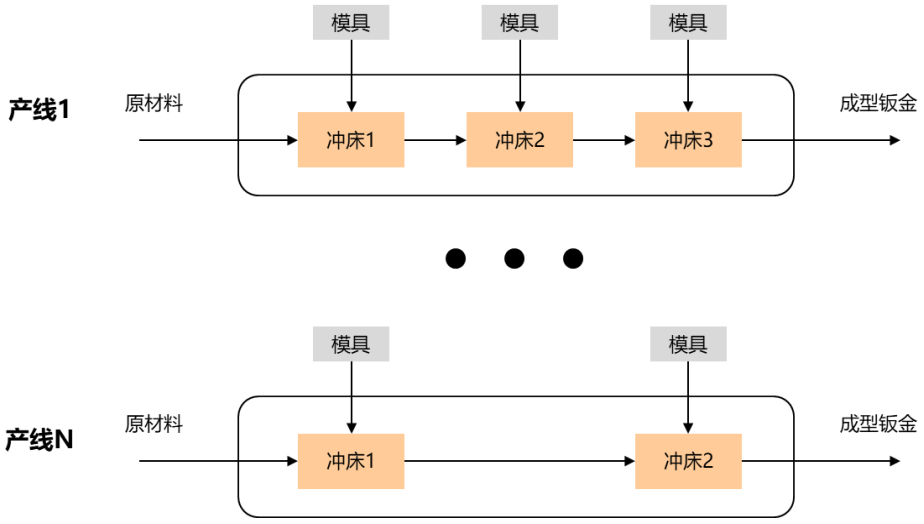

图4.26 钣金成型加工工序示意图

钣金车间根据不同的工厂管理制度以及淡旺季需求，通常会切换多种类型班次，班次内分别由非加班工时和加班工时组成，其中工作日和节假日的模式也会有所区分。钣金加工计划需决策班次安排（标准/灵活模式）、人员分配及各产线生产数量，目标为最小化人力成本、时间成本、库存成本、未满足需求成本及需求成本，其中通常未满足需求成本优先级最高。

在钣金月度或周度计划指导下，车间需进一步安排钣金的具体加工任务，包含钣金类型、数量以及具体的排产时间。需要考虑实时的库存水位、模具可用性、人员可用性、原材料等因素，充分利用产能，减少等待、整体加工时间和库存积压。

4.8.1.2 业务痛点

钣金计划与排程的主要业务痛点如下：

1) 资源协同复杂：需平衡产能（产线加工时长与班次时长）、人力（工人流动性大）、原材料（短期库存限制）及库存容量，传统排产依赖人工经验，难以应对多约束协同；

2) 换型成本高：模具更换导致生产启动成本显著，且未满足需求成本优先级高，需优先保障关键订单，但换型策略缺乏系统性优化；

3) 班次模式选择：标准模式（周末无加班）与灵活模式（周末可安排晚班）需动态调整，其中周末班次时长可灵活配置，协调难度大；

4) 缺货风险管理：当排产冲突时，需按规则优先级（原材料库存→产能→库存→班次模式）调整，但规则冲突处理逻辑复杂，缺乏系统化支持。

4.8.1.3 技术挑战

钣金计划与排程的算法实施的技术挑战如下：

1) 动态调整与多目标平衡：需在最小化成本（人力、库存）与满足需求（优先级高）间找到最优解，且需应对突发需求变化（如临时订单调整）；

2) 规则冲突与二次排产：当首次排产无法满足需求时，需按规则优先级调整，但冲突处理逻辑复杂，需明确输出违规详情（如某产线超负荷运行）；

3) 算法求解效率：面对多产线、多班次、多约束的组合优化问题，传统方法难以在合理时间内生成可行方案，需依赖高效算法支持；

4) 后工位衔接管理：成型后的钣金需交付至点焊工位，需预留缓冲时间，避免因前序延迟导致后工位等待，但缓冲策略需与排产逻辑协同设计；

5) 每条产线的运行时间（如班次类型，是否加班、周末是否开工等）的决策复杂度高，对于成本和可用人工的影响很大，需要进行系统性的规划。

4.8.2 业务需求

4.8.2.1 业务场景目标

钣金加工业务场景需针对不同钣金加工任务（类型、数量及精确交期）制定月度排程，考虑产线换型时间（不同顺序影响效率）及产能、工人数、原材料库存约束[149][150]。核心挑战在于生产时长与工人数为决策变量而非固定参数，导致变量交互复杂、决策维度高，需在满足交期的前提下优化资源分配，但当前缺乏成熟方案，需创新性解决多目标协同优化问题。

在细排程中给定钣金的加工任务（类型、数量）、每筐钣金的交期（具体到分甚至秒），需要考虑产线从一种钣金切换到另一种钣金任务时换型，换型会占用产线时间率。

4.8.2.2 月度计划目标和功能约束

（一）集合和常量

钣金月度计划场景涉及的关键数据集合和常量如下：

1) 钣金信息：每个钣金半成品的需求、现有库存、安全库存、目标库存要求以及可生产机器和效率等基础产品信息或业务数据；

2) 产线信息：包含产线编码、机器在产钣金状态等信息，以及每个机器在不同时间的最大产能；

3) 辅助资源信息：包含模具限制等，以及模具与产品、机器的关系；

4) 成本信息：包含优化目标中的成本参数，包含库存成本、换型成本、缺货成本、班次成本参数；

5) 班次信息：包含可以使用的班次，以及每个班次的常规工作时间和可加班时间。

（二）月度计划优化目标

表4.11列举了月度计划业务模块的主要业务目标。

表4.11 月度计划业务目标

目标	说明
人力成本	每条产线、每天的人力成本求和（时间单位为秒）

产能成本	班次总时间尽可能小，并考虑到加班成本系数增加
库存成本	每类钣金、每天高于安全库存的成本与低于安全库存的惩罚成本
需求未满足成本	有某天需求未满足发生时的惩罚成本

（三）月度计划场景功能和约束

表4.12介绍了月度计划场景的关键约束。

表4.12 月度计划业务约束

类型	约束	说明
资源	最大产量	单班次最大产量
	模具数量	有模具才可生产，多个钣金共用一套模具
	产线/模具维护	产线/模具维护期间不能生产
环境	库存	库存、生产量、需求和延期需求平衡；库存容量
物料	原材料库存	原材料数量限制
人员	人力资源	可用人力限制
	班次时间	生产时间限制

4.8.2.3 现场排程目标和功能约束

（一）集合和常量

钣金月度计划场景涉及的关键数据集合和常量如下：

1) 钣金信息：每个钣金半成品班次内的月度计划需求量、现有库存、目标库存要求以及在可生产机器上的加工时间等基础产品信息或业务数据；

2) 产线信息：包含产线编码、机器在产钣金状态等信息，以及每个机器在不同时间的最大产能；

3) 辅助资源信息：包含模具限制等，以及模具与产品、机器的关系；

4) 成本信息：包含优化目标中的成本参数，包含库存成本、不同产品间的换型成本、延期成本、安全库存成本参数；

5) 班次信息：包含可以使用的班次，以及每个班次的常规工作时间和可加班时间。

（二）钣金现场排程优化目标

表4.13介绍了钣金现场排程的优化目标需求。

表4.13 现场排程业务目标

目标	说明
换型时间	换型总时间尽量小
库存	库存尽量小

（三）钣金现场排程模块提供的主要功能

表4.14列举了钣金现场排程的关键场景功能和约束说明。

表4.14 钣金现场排程场景业务约束

类型	约束	说明
人员	人员配置	产线班次人员配置，排产钣金不能超过所需人员数目
产能	产线可生产时间	正常班次，除去故障、维护时间
	模具可用时间	模具维护期间无法生产对应钣金
	共模具约束	多种钣金共用一套模具，当一种钣金占用模具时，其它钣金无模具可用。
	钣金交期	满足钣金交期约束
环境	库存	满足最大库存约束

4.8.3 技术方案

4.8.3.1 技术方案设计

针对钣金加工面临的多目标平衡、多规则约束及备库生产等挑战，华为天筹智能排产引擎提供了一套全面的技术方案（见图4.27）。该方案通过对系统的实时信息进行快速排产，实现了从生产管理科到制造员的全流程协同优化。

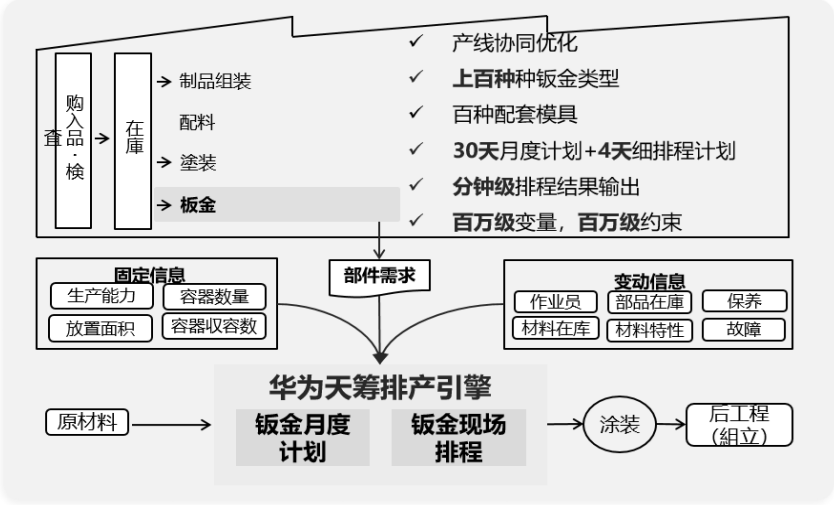

图4.27 钣金加工计划和排程场景解决方案

系统基于订单交期、工艺、产线产能、模具保养、人员配置等复杂规则，生成月度计划和4天细排程，确保生产任务的精确安排。其次，通过百万级变量和约束的优化计算，实现多条产线的协同运作，涵盖百余种钣金类型和套模具，确保生产计划的科学性和可行性。

此外，系统支持分钟级排程结果输出，实时调整生产计划以应对突发情况，提升响应速度。在换型时间优化方面，系统通过智能算法，根据不同加工顺序动态调整换型时间，提高产线可动率。同时，系统还考虑了原材料库存和工人数的动态变化，确保生产资源的合理配置，降低库存占用和资金压力。

4.8.3.2 算法框架设计

算法整体包含上下两层，上层求解批量生产计划问题[151]，下层求解带机器族的单机调度问题[152]。算法流程如图4.28所示，包含了并行批量生产计划和产线顺序依赖调度排程，分阶段进行迭代优化。

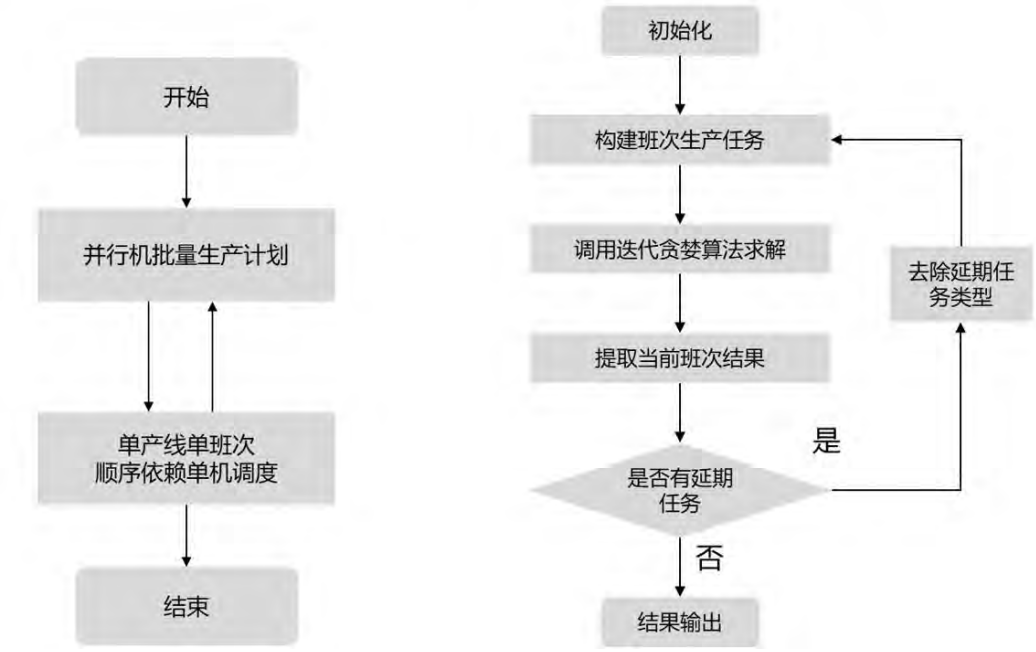

图4.28 钣金加工计划与排程算法示意图

4.8.4 应用效果

通过华为天筹智能排产引擎在实际应用中取得了显著成效：

1) 通过多目标综合考虑和多约束条件优化，系统成功降低了生产部件库存20%；

2) 整体产线产能利用率提升2%；

3) 算法引擎分钟级输出结果，可根据动态参数调整实时刷新方案，及时响应突发事件，确保生产计划的灵活性和适应性。此外，系统通过精确到产线的排产安

排，有效解决了备库生产导致的高库存问题，实现了资源的高效利用。

4.8.5 复制推广建议

该方案的可复制性与推广性主要体现在其解决的复杂排产与资源优化问题，适用于以下行业、领域或企业类型。

1) 家电制造行业（如冰箱、洗衣机、空调等）

适用场景：钣金件是家电的核心结构件，需应对多型号、小批量、高频换型的生产需求。具有多产线协同排产（如空调外壳、洗衣机底座等不同零件的模具冲突），淡旺季需求波动导致的班次模式切换和较大换型时间损失的场景。

2) 电子设备制造

适用场景：电子设备订单需求碎片化，对于交期敏感，且产线资源动态调整较多，宜推广包含多品种小批量的电子设备制造企业。

3) 工业设备与通用机械制造

适用场景：工业设备通常涉及较多定制化组件需小批量生产，需要多产线协同生产满足不同客户订单需求。定制化订单导致的工艺路线差异，对产线产能与订单优先级的动态匹配要求较高（如紧急订单插单处理）。通过高效的智能算法动态调整排产结果，提升需求变化和异常响应效率。

4) 医疗设备制造

适用场景：高精度医疗部件（如手术器械支架、设备外壳）通常需小批量生产，具有严格的质量管控与工艺合规性要求。涉及多工艺路线（如折弯+焊接+表面处理）的协同排产较多，小批量订单导致模具利用率低，且交期敏感（如医院紧急采购需求）。通过计划排程优化算法可以减少换型频次并提升产线利用率。

4.9 医药物流配送技术方案与应用实践

4.9.1 业务背景介绍

4.9.1.1 业务场景描述

从供应链整体视角看，医药物流配送不是孤立的“送货”，而是贯穿采购-生

产-分销-终端的全链条协同；通常可划分以下三个阶段：

1) 上游供应与生产物流阶段：该阶段是药品从“原料、辅料、包材采购进厂，再到厂内生产流转，最终到成品入库待发”的全过程，是整个医药物流供应链的质量与合规起点。主要包括供应商到生产企业仓库的采购物流，以及原料库到车间，再到成品库的厂内生产物流。

2) 分销调拨物流阶段：该阶段是药品从生产企业/总代，通过一级、二级商业公司、区域配送中心，进行批量调拨、分拨、拆零、集货，最终形成面向终端（医院、药店、诊所）的供应网络的主干物流环节，以多级分销、跨区域调拨、批量与拆零并存为特征，承担库存调度与网络覆盖功能，涉及干线运输、区域分拨、库间调拨等场景。

3) 终端配送物流阶段：如图4.29所示，终端配送是药品从区域配送中心/商业公司，通过城市配送、落地配等方式，精准送达医院、药店、诊所等终端使用场景的“最后一公里”交付环节，核心是实现小批量、多频次、多温区、高时效的精准交付，是医药物流与终端需求直接对接的闭环节点。

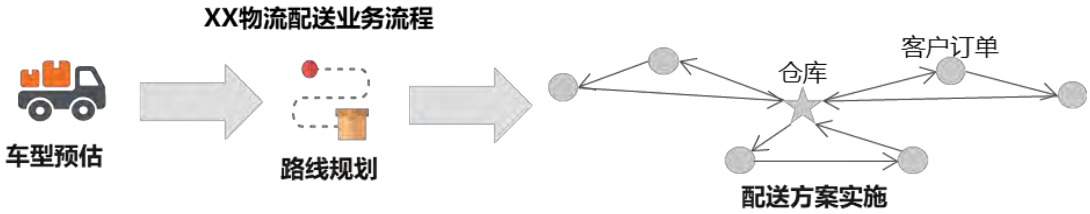

图4.29 终端物流配送业务示意图

其中，终端配送物流成本在全供应链中占比25%-40%，是成本权重最高的环节之一，其高占比本质是作业碎片化、场景多元化等复杂度维度的体现，这一占比虽因场景差异有所波动，但通过网络优化、技术赋能与规模化运营，可实现明显的降本增效。一般来讲，终端配送物流均可抽象为已知仓库、已知多配送点需求、多车场、多车型供给条件下的配送路线优化问题[153]。

4.9.1.2 业务痛点

当前医药终端物流配送存在如下痛点：

1) 业务体量大，传统管理模式难以匹配规模化运营的精准调度需求；以XX医药配送公司终端配送为例，6000+拆零品类覆盖广泛需求，日均 500+订单、5K 件

出库量构成高频作业基数，叠加150+可用车辆调度与50+仓内人员协同，形成“多品类、高订单、大流量”的复杂运营场景。

2) 运营效率低，流程断点与经验依赖拖慢周转；主要表现为司机凭经验选货排线、车辆装货缺乏标准化流程，不仅延长单票作业周期，更可能导致配送延误，整体拉低供应链周转效率。

3) 运营成本高，配送路径依赖人工经验，未实现最优路径规划；主要表现为里程冗余、燃油与时间成本较高，叠加人员、车辆的低效运转，形成“高成本、低产出”的恶性循环。

4) 智能化程度低；从发货计划、仓库补货，到拣货库位分配、任务指派、打包箱型选择、路线优化，全流程依赖人工经验决策，缺乏数据驱动与智能算法支撑。

4.9.1.3 技术挑战

当前医药终端物流配送存在如下技术挑战：

1) 交付窗口碎片化，路线合并难，规模化效应弱。不同终端客户的“必须送达时段”分散，导致路线无法集中合并，反而需拆分多趟配送，空驶率与运输成本上升。

2) 临时订单与异常事件，导致动态调整难，调整成本高。终端客户临时加急订单、配送途中异常（如车辆故障、药品破损）、城市货车限行（如某路线内白天禁行）、高峰期拥堵、临时交通管制等突发情况，会打乱原规划路线，导致后续订单延误、空驶率上升；排线系统缺乏高效响应机制和计算能力。

3) 终端卸货与验收效率差异，导致路线衔接不畅，整体时效拖慢。不同终端的卸货、验收耗时差异极大，导致路线规划的“时间预估”偏差，后续终端配送延误，连锁影响整个路线的优化效果。

4.9.2 业务需求

4.9.2.1 客户整体功能需求

针对医药终端配送物流，客户希望通过算法工具替代人工依赖，实现排线、装箱、调度的标准化、智能化运作，同时精准核算配送成本，提升配送效率、降低误

差率与运营成本。整体需求如图4.30所示，具体描述如下：

1) 支持智能化自动排线：统筹当日订单、收货点数据，结合“收货时间窗、总里程最短、用时最少”约束，自动生成最优排线方案，替代人工排线。

2) 支持最优车型推荐：系统结合排线结果、车型参数及装载率要求，自动推荐匹配车型，替代人工预估。

3) 支持提供装车指导：系统输出提前规划的装车方案，提升装载效率与装载率稳定性。

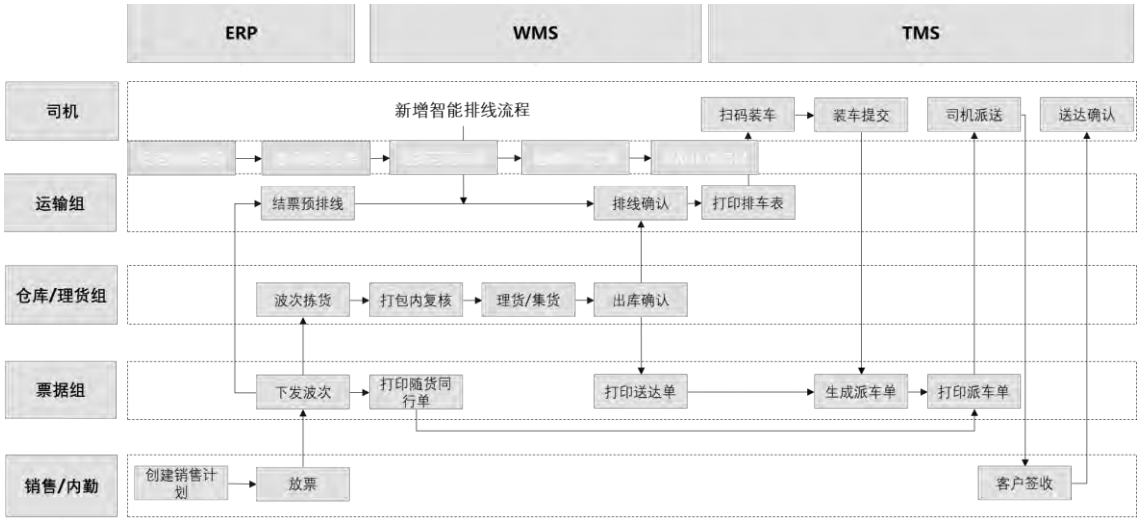

图4.30 智能排线需求示意图

4.9.2.2 集合和常量

医药终端配送涉及的关键数据集合和常量如下：

1) 订单信息：订单编号、客户编号、配送数量、配送体积、配送重量等；

2) 车辆信息：车辆类型、可用数量、可达客户编号、载重上限、容量上限等；

3) 路网信息：两两节点间的配送距离、耗时数据等。

4.9.2.3 业务目标

表4.15为医药终端配送的业务目标概述。

表4.15 医药终端配送业务目标

目标	描述
最小化配送里程	所有配送订单执行完毕后，车辆执行里程的总和最小
最小化发车车次	整体排线结果尽量满足使用车次最少

4.9.2.4 业务约束

表4.16为医药终端配送的业务约束概述。

表4.16 医药终端配送业务约束

约束	说明
装车原则	按收货点的排线顺序，遵循 先装后卸，后装先卸的装车原则
单车路线最优	单个车次的配送顺序尽量满足线路最优
时间窗要求	客户有收货时间窗要求，到达客点的时间必须在给定的时间范围内
单车程行驶时长要求	结合卸货时长，单车车次的去程和回程总时间有要求
订单拆分配送需求	部分订单存在一车装不下的情况，需要算法自动拆单
多仓库需求	出库库区按货物品类有多个：例如毒麻药品、冷链（部分跟普药一起）、普药（器械、中药）
同客户订单连续配送要求	同一个客户的订单尽量在一个车上配送完毕

4.9.3 技术方案

基于启发式、元启发式、整数规划等方法，构建并自适应选择多种类的数学模型[154][155][156][157][158]，图4.31为对应技术方案示意图，具体实现路径如下：

1) 模型构建：明确决策参数（订单信息、位置信息、车辆信息、路网信息等）建立满足客货匹配、车货匹配、路网匹配等多维度约束条件的数据模型，并以“提高车辆利用率、降低运费成本”为核心目标，构建目标函数；

2) 模型求解：借助天筹 AI 求解器等专业求解工具，对构建的数学模型进行高效求解，确保求解结果的最优性与时效性；

3) 方案输出：根据求解结果，生成符合实际业务需求的排线方案，明确各车辆、订单的配送方案、计划发出时间等关键信息。

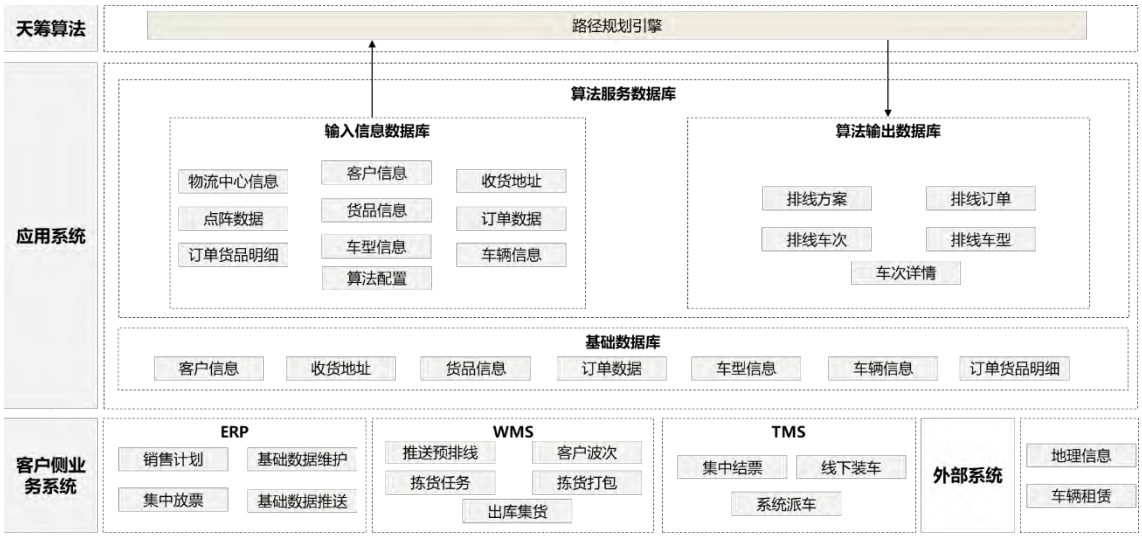

图4.31 医药终端配送解决方案

4.9.4 应用成效

通过上述技术方案，在满足业务需求的同时，同步有如下应用成效：

1) 分钟级级结果输出能力：单次排线耗时从1h手排缩短到算法60s内完成求解；

2) 提升业务效果：运输成本减少约15%；

3) 泛化性高：规则可配置，支持同城、跨区域配送等多种场景。

4.9.5 复制推广建议

具有路线优化需求的供应链物流场景，核心应用领域包括：

1) 厂内物流：适配客户工厂/仓库内部的全流程物流服务，涵盖原料接收、仓储管理、产线配料、成品入库及装运等环节。厂内物流普遍面临“多节点、短距离、高频次”的路线调度需求，可优化原料到产线、成品到仓库的转运路线，解决厂内车辆拥堵、人员协同低效等问题，尤其适用于品类繁杂、产线密集的制造型企业。

2) 干线物流：聚焦连接区域中心、城市或港口间的长距离、大批量货物运输场景。干线物流的核心痛点是“里程长、成本高、运力调度难”，可结合实时路况、货物时效要求、车辆载重等因素，规划最优干线运输路线，减少无效里程与燃油消耗，同时优化车辆调度与配载方案，解决传统干线物流中“空驶率高、调度滞后”的问题，适配电商平台、大宗商品贸易等需要跨区域运输的行业。

3) 终端配送物流：针对连接个人与商家的末端配送环节，覆盖快递、社区团

购、生鲜配送等环节。终端配送面临“订单分散、配送点多、时效要求高”的特点，可结合配送地址聚类、订单优先级排序、实时路况更新等，优化末端配送路线，减少配送员无效折返，提升单趟配送效率，同时降低配送里程与时间成本，适用于连锁零售、生鲜电商、即时配送等对末端履约效率要求高的领域。

4.10 服装智能排唛优化技术方案与应用实践

中国是世界最大服装生产国和出口国，生产工厂主要区域分布于劳动力集中和产业链完善的地区，如珠三角、长三角地区。服装生产制造业是包含设计、裁剪、缝纫、后整及包装的劳动密集型工业，也是中国重要的国民经济支柱之一。其中，面料成本占服装生产成本的40%以上（梭织、高端面料占比更高），裁剪环节的布料损耗是服装生产中最大的可控损耗来源，排唛算法正是通过解决这一核心痛点，从成本、效率、质量、运营、产业升级多维度为服装生产制造创造价值，同时适配当前服装行业“多品种小批量”、“小单快反”的核心产业趋势，成为现代服装工厂柔性供应链的必备核心技术。

4.10.1 业务背景介绍

4.10.1.1 业务场景描述

1) 服装生产基本流程如图4.32所示：先经设计师设计打版出确认样，然后面辅料入厂检验，随后进行排版、裁剪、缝制、整烫成型，成品检验合格后按标包装，最后完成仓储或物流交付。面料成本占服装生产成本的40%以上，排料效果的优劣对企业生产成本有重要影响。

2) 服装排版是服装裁剪环节的核心前置工序，是指在满足设计、制作等要求的前提下，将服装的所有裁片在面料指定幅宽内进行排列，以最短使用长度给出排版图，使面料的利用率达到最高，排版性能的优劣直接影响布料成本。

3) 服装排版引擎在20世纪90年代即在欧美国家开始研究，经过长时间迭代发展，排版引擎的生产约束由人工经验转换成引擎特性，行业定制化少。顶级引擎大部分来自国外，国内厂商能力主要集中在CAD软件和生产流程管理，通过外采排

唛引擎licence提供服务。

4) “小单快反”是服装行业适配快时尚趋势的柔性生产模式，以小批量起订、短交期交付、可快速返单为特征，是目前跨境电商、快时尚品牌、设计师品牌的主流合作模式，需要全链路的柔性化和数字化能力。

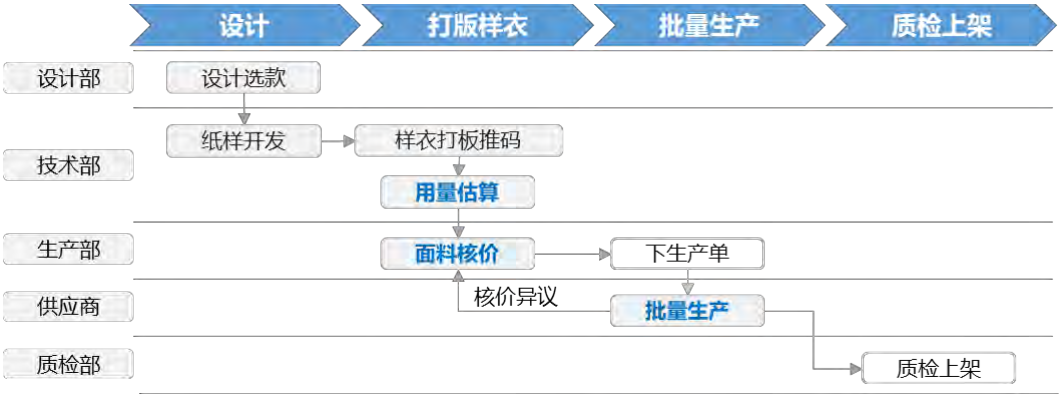

图4.32 服装生产业务流示意图

4.10.1.2 业务痛点

1) 手工排版耗时耗力性能低：传统手工排料方法严重依赖人工经验，需要经验丰富的排版师傅花费40~60分钟才可以完成一次排版任务，面料平均利用率仅在70%左右，不仅耗时耗力，面料利用率水平也比较低，造成人力物力的双浪费；

2) 线下软件更新适配难：线下软件迭代周期长，需要部署调试硬件，无法根据生产的淡旺季进行动态扩缩容，且仅能在本地或者局域网使用，远程协作效率低，制约行业柔性供应链生产效率。

4.10.1.3 技术挑战

1) 要求在满足生产约束的条件下，相比友商的性能提升0.6%，且持续进行料率提升，同时排料劣于友商的数据比例控制在12%以下；

2) 要求公有云服务稳定性，可服务水平保证在99.99%，且提供动态扩容能力，以应对服装生产淡旺季。

4.10.2 业务需求

4.10.2.1 客户整体功能需求

客户对服装排唛算法的功能需求主要有如下三点：

1) 高效自动排唛算法：减少对人工经验的依赖，缩短单次排版耗时，同时提升面料利用率，达到降本增效的目的。

2) 排唛算法敏捷迭代，弹性适配生产淡旺季：摆脱线下本地软件迭代慢、需硬件部署调试的局限，支持算法版本快速迭代更新，可根据生产淡旺季动态扩缩容，降低部署与维护成本。

3) 支撑远程协作，适配小单快反柔性生产：解决远程协作效率低的问题，满足服装行业 “小批量、短交期、快返单” 的小单快反模式，具备全链路数字化与柔性化能力，适配快时尚、跨境电商等柔性供应链需求。

4.10.2.2 集合和常量

服装排唛算法涉及的关键数据集和常量如下：

1) 面料信息：包括面料的尺寸、条纹等信息；

2) 裁片信息：包括裁片轮廓、数量、旋转角度、尺码信息等；

3) 算法运行时间：该常量由人工设置，适配不同的排唛场景。

4.10.2.3 业务目标

服装排唛算法核心业务目标是在满足生产约束的前提下最大化面料利用率，提升排唛自动化效率，缩减排唛耗时，最终实现面料降本、生产提效，助力服装企业降本增效和柔性化生产落地。

4.10.2.4 业务约束

服装排唛是面料裁剪前的核心环节，需要严格遵循面料特性、生产设备、工艺标准、订单要求等多重生产约束，如表4.17所示，具体核心约束可分为面料、裁片和生产约束3大类。

1) 面料约束

a) 布料形状：布料幅宽固定，要求排版裁片的总宽度不得超过面料幅宽；

b) 避瑕疵约束：排版时需避开面料上的破洞、污渍、跳纱等疵点，避免疵点裁入裁片导致成品不合格；

c) 锁定排料：排版时将某裁片固定在面料位置上，以便保证面料图案固定在特定裁片上。

2) 裁片约束

a) 裁片形状：支持任意简单多边形的异形裁片形状；

b) 裁片旋转：由于面料由经纬线组成，支持裁片可以旋转90度和180度两种角度；

c) 裁片翻转：由于面料表里区别，支持裁片可以沿x、y轴翻转；

d) 组合排料：两个或多个裁片预先组合在一起，排料结果中相对位置保持不变；

e) 同套同方向排料：相同尺码的裁片的旋转和翻转方向保持一致。

3) 生产约束

a) 裁片间距：裁片之间预留间隙余量，避免裁剪时伤到裁片；

b) 分行排料：不同套裁片放置面料上下不同区域；

c) 分列排料：不同套裁片在面料的左右不同区域。

表4.17 二维切割求解引擎特性说明

序号	生产约束	详细说明
1	布料形状	支持矩形布料
2	裁片形状	支持矩形以及异形裁片
3	裁片旋转	裁片90、180度旋转
4	裁片翻转	裁片翻转
5	裁片间距	裁片之间间隔相等距离
6	锁定排料	裁片固定在面料上
7	组合排料	控制多个裁片之间的位置关系
8	避瑕疵排料	自动避开布料上设定的瑕疵区域
9	分行排料	不同套裁片在面料上下不同区排版
10	分列排料	不同套裁片在面料左右不同区排版
11	同套同方向排料	同套裁片有相同的旋转角度

4.10.3 技术方案

天筹依托运筹学经典启发式算法与整数规划方法为核心技术底座，融合高效计算几何算法，精准处理裁片轮廓、尺寸、间距等几何关系，确保排唛方案完全适配面料幅宽、裁片纱向等生产约束，避免因排版偏差导致的裁片报废、成品瑕疵；同时嵌入AI加速技术，通过机器学习迭代优化排唛策略，结合历史排唛数据自主适配不同面料特性、裁片类型的排版需求，大幅提升排唛效率与方案精准度，最终构建出一套既能满足服装排唛生产约束，又能兼顾性能的算法引擎。

为降低行业使用门槛，天筹服装排唛引擎采用公有云服务形态（见图4.33），无需企业投入本地服务器、IT运维团队进行部署维护，实现“开箱即用”。同时，联合服装行业上下游生态伙伴，整合排唛前后端相关服务资源，共同打造从服装设计、打版推码、裁片排版到对接裁剪设备的云端一体化排唛服务，打通排唛环节的技术壁垒，助力企业简化排版流程、降低运营成本。品牌方可通过自身系统汇聚全渠道排唛任务，统一调用天筹公有云排唛API，快速获取符合生产约束的排唛方案；工厂则可通过生态伙伴客户端同步接收品牌方下发的排唛任务，直接对接裁剪设备开展生产，实现全流程数字化联动。

图4.33 服装排唛解决方案

4.10.4 应用成效

天筹服装排唛引擎当前已成功上线公有云，多个用户调用服务用于服装切割排版，为用户提供高效的解决方案，降低用户布料成本。

1) 求解性能高效

通过系统性算法优化，天筹服装排唛算法性能超越国外顶级引擎，在相同的求

解时间内，在常见的排版场景里，天筹引擎平均布料利用率相比友商提升0.6%。

2) 支撑特性全面

天筹服装排唛引擎支持行业常见的生成约束，包括分行分列、同套同方向、裁片有限旋转和固定排料等，满足绝大部分客户的排唛诉求。

3) 全链路供应链协同

实现服装柔性供应链品牌方和工厂全链路的数字化能力升级，提升供应链响应速度，降低协同成本。

4) 产业价值突出

服装排唛是服装生产中的关键环节，排唛效果优劣影响最终的服装生产成本。根据行业经验估算，2024年全国服装生产面料成本约X000亿元，面料利用率每提升0.1%，总体面料成本将降低数亿元。

4.10.5 复制推广建议

建议推广至服装行业年销售额在5亿+的服装生产企业，相关地域规模以上企业如表4.18所示。

表4.18 全国规模以上服装生产企业地域分布数量预估说明

地域	企业数	年销售额（亿）
浙江	10	799
福建	7	837
江苏	6	607
山东	5	238
上海	4	230
北京	3	451
广东	3	109
内蒙古	1	33

4.11 钢铁企业智慧配煤技术方案与应用实践

4.11.1 业务背景介绍

炼焦配煤是钢铁生产中决定焦炭质量与成本的关键环节。简单来说，它是将多种性能迥异的单种煤（如气煤、肥煤、焦煤等）按比例混合，通过焦炉高温干馏制

成焦炭的过程。配煤优化则是指综合考虑现有库存、原料煤特性、采购成本及成品焦炭指标等因素，确定煤种的最优配比。企业需在满足高炉对焦炭热强度和反应性要求的前提下，动态调整配方以消纳长协煤或低价替代煤，对冲原材料市场波动，降低生产成本。

4.11.2 业务需求

针对复杂多变的工业环境，企业对配煤决策的诉求集中在以下三个核心维度：

1) 成本控制：在确保最终产品质量达标的前提下，最大程度地提高低价原材料的使用比例，降低整体配方成本；

2) 质量稳定性保障：克服原材料（如原煤的灰分/硫分）批次间属性波动大的难题，确保最终产成品关键质量指标的稳定，减少次品率；

3) 决策经验数字化：将依赖少数资深配方师的隐性经验转化为可标准化的数字模型，避免因人员流动导致的生产波动。

尽管场景的优化目标明确，但在实际工业生产中实现最优配料面临着极高的复杂性，传统的基于人工经验或 Excel 计算已难以应对，主要业务挑战包括：

1) 经验依赖：炼焦配煤过程极其复杂。对于部分指标，存在明确的物理/化学机理公式（白盒）；但对于许多关键工艺指标，原材料与产成品之间的关系是非线性的、强耦合的，缺乏明确的机理公式支撑（黑盒）。传统单一的优化手段无法同时解决这两类问题；

2) 原料属性的动态变化：上游原材料供应源复杂，不同批次的原料属性可能发生显著变化。当新批次原料入库时，原有的最优配方可能失效，需要系统具备分钟级的动态重算能力；

3) 多维约束的组合爆炸：一个典型的配方决策需要同时满足数十项硬性约束，包括质量下限、环保指标上限、特定原料库存可用性、设备产能限制等。在巨大的解空间中寻找同时满足所有约束且成本最低的帕累托最优解，常常陷入组合爆炸困境，人工决策不仅耗时且难以保证最优性。

4.11.3 技术方案

配料优化是工业生产企业普遍存在的业务需求，在钢铁、能源、化工及大型食品加工等行业中都有相似的场景。天筹团队针对配料优化场景提出了一种“机理与数据融合驱动的混合智能配料优化系统”，该方案的核心创新在于其独有的“双模态优化引擎”，能够根据业务场景的机理明确程度，自适应地选择最优求解路径。该系统并没有强制要求所有业务场景都具备完美的机理模型，而是提供了两种并行的解决路径：

路径1：基于数学规划的精确求解。对于成本、基础化学成分等具有明确线性或凸规划机理公式的场景，系统将业务问题建模为混合整数线性规划（MILP）或非线性规划（NLP）模型。通过集成业界领先的天筹数学优化求解器，在严格满足所有硬约束的前提下，快速求得理论层面成本最低的精确解。

路径2：基于“代理模型+黑箱优化”的寻优。对于缺乏明确机理公式的复杂质量指标，系统首先利用历史生产数据，训练高精度的机器学习模型（如华为预测大模型）作为代理模型用于模拟和预测配比结果。随后，运用天筹黑箱优化求解器，在巨大的配方空间中高效搜索，通过不断与代理模型进行交互迭代寻优来逼近最优配比方案。

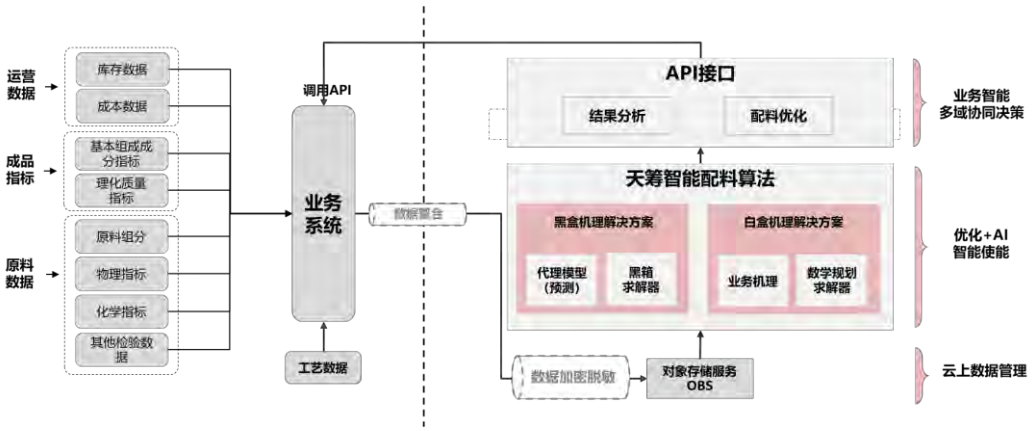

图4.34 智能配料解决方案

4.11.4 应用成效

以上智能配料优化系统在多个头部能源、制造企业落地，在焦炭配煤、烧结配

料、面粉比例调配等场景，实现了从经验决策到数字化智能决策的跨越，取得了显著的收益。以某钢铁企业的智慧配煤场景为例，系统能够在严格的质量公差约束下降低配合煤成本约5-20元/吨，按照年产200万吨焦炭计算，可节省成本上千万。此外，系统将单次配方设计耗时从数小时缩短至2分钟以内，整体决策效率提升逾90%。这种敏捷的响应能力使企业能够从容应对原材料市场的剧烈波动，在确保生产连续性的同时，实现了资源利用率与核心竞争力的双重跃升。

4.11.5 复制推广建议

配料优化方案的普适性极强，广泛存在于那些需要将多种原材料按比例混合以达成特定性能或成本目标的工业场景中。在钢铁行业，它体现为炼焦配煤或烧结配料的成本压降；在能源领域，它是火电厂为平衡排放与热值而进行的煤种掺配；而在食品与畜牧业，它则是针对大豆、玉米等原料波动进行的动态饲料配方生成；在建材行业，水泥、混凝土等基础建材的组分设计也需要配料优化技术。在推广过程中，应该重点关注那些原材料成本占比高、配方调整频繁且质量约束严苛的流程制造企业，这些通常是决策智能最能产生价值的场景。

“白盒”场景：针对机理明确、成分比例呈线性叠加关系的“白盒”场景，推广的重点在于通过数学规划实现极致的成本最优与决策自动化。此类方案的成功落地高度依赖于企业基础数据的规范性，因此销售人员在介入前期需确认客户是否具备完善的实验室信息管理系统（LIMS）或ERP系统，以保证原料的化学成分、库存余量及实时价格能被实时采集并自动同步。此外，业务规则的数字化程度也是关键，企业必须能够清晰定义各项质量指标的上下限约束以及环保等政策性硬约束，只有当这些“显性规则”被充分梳理并转化为数学语言后，白盒优化引擎才能在确定的解空间内精确求解最优配比。

“黑盒”场景：而在面对生产机理复杂、产成品指标与原料配比呈现非线性强耦合关系的“黑盒”场景时，推广策略则应转变为“数据驱动+专家经验”的数字化转型升级。由于此类场景缺乏明确的物理化学公式支撑，销售人员应优先识别那些数字化转型意识较强、且积累了至少两年以上高质量历史生产全要素数据的企业。前期准备工作的核心在于数据治理的完备性，包括传感器采集的工艺参数、原材料属性以及最终产成品的化验指标是否实现了闭环关联。在这种模式下，方案的

价值点在于利用机器学习构建代理模型来替代人工经验进行指标预测，因此企业是否具备支撑高频模拟预测的计算环境，以及是否愿意配合算法团队进行模型迭代与反馈，将直接决定黑盒优化方案能否真正从实验室走向生产线并实现价值闭环。

4.12 钢铁企业余气发电优化技术方案与应用实践

4.12.1 业务背景介绍

在钢铁工业的价值链条中，能源成本通常占据总生产成本的20%至30%，是影响企业核心竞争力的关键变量。大型钢铁企业通常具备复杂的二次能源系统，包括高炉煤气、焦炉煤气及转炉煤气等副产资源的回收与再利用。随着电力现货市场的交易机制日益完善，电价的实时波动对企业自备电厂的调度灵活性提出了更高要求。在原材料价格处于高位的行业背景下，钢铁企业亟需通过提升能源系统的调度精度，将光伏、风电等波动性新能源与传统二次能源进行深度耦合，以实现厂区内部微电网的供需平衡。这种从传统的“保障生产”向“经济运行”的职能转变，已成为钢铁企业构建动态成本优势、达成低碳转型目标的客观路径。

4.12.2 业务需求

钢铁生产流程具有强耦合、大惯性及多变量非线性等特征，这导致能源管理部门在实际调度中面临多维度的复杂约束。首先是煤气系统的动态平衡诉求，由于高炉、转炉的工况会随生产节奏频繁波动，煤气产生量与消耗量难以完全同步，极易引发两类问题：一是气柜柜位超限导致煤气放散，造成能源浪费；二是柜位过低引发生产安全风险。其次是复杂电价结构下的决策博弈，企业需要在保障厂区生产用电的前提下，结合峰平谷电价实时调整自备电厂的出力与储能系统的充放电节奏。此外，调度系统还须兼顾发电机组的爬坡速率、气柜压力稳定以及储能电池寿命等物理约束。传统的依靠调度员经验的人工模式，在处理跨时间尺度的多目标寻优问题时存在计算边界受限与决策迟滞等瓶颈，难以在复杂的约束条件下实时捕捉全局最优解。

4.12.3 技术方案

针对钢铁场景的能源调度特征，本方案采用了机理模型与数据驱动相结合的混合建模架构。在逻辑清晰且遵循能量守恒定律的环节，方案构建了基于混合整数线性规划（MILP）的数学优化模型，将发电机组出力、气柜容积变化、储能状态等核心指标设定为决策变量。模型严格集成了生产安全硬约束，包括煤气压力波动范围、机组物理爬坡限制以及电力合同申报量等边界条件，确保调度指令的物理可行性。针对煤气产消预测等具有极强非线性且受多工况影响的环节，方案引入了基于深度学习的代理模型，通过对历史生产数据的特征提取，实现对未来24小时能源供需趋势的高精度前瞻。这种“白盒”寻优与“黑盒”预测深度耦合的架构，使决策引擎能够在复杂的物理约束下，自适应地应对生产波动，实现能源分配的动态帕累托最优。

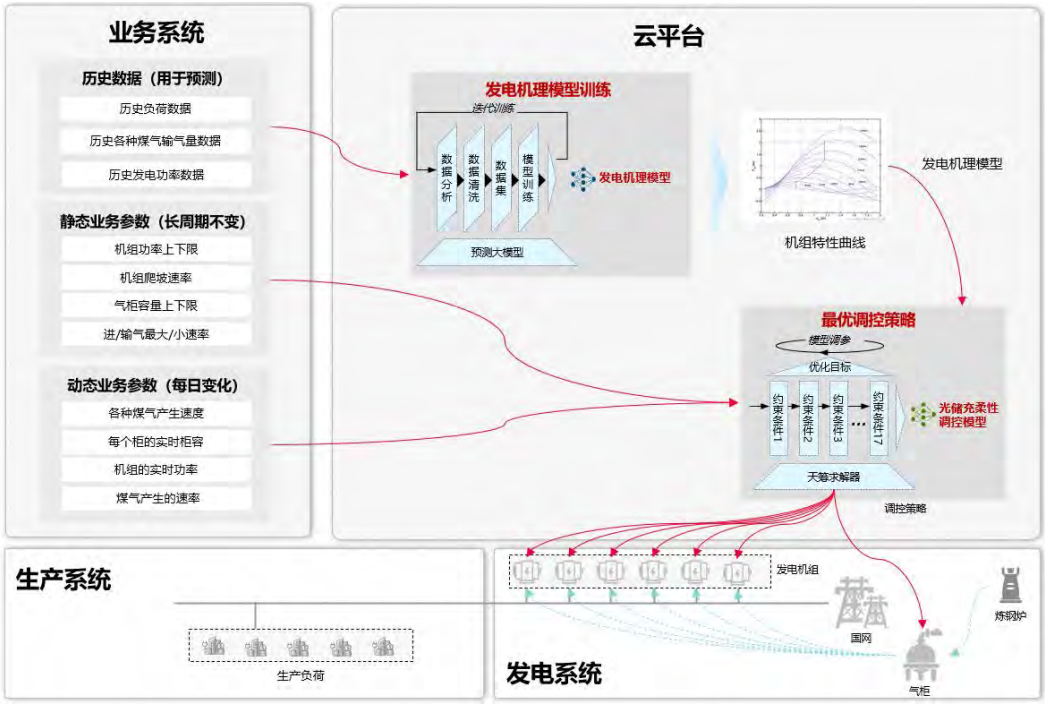

图4.35 钢铁余气发电优化技术架构

4.12.4 应用成效

通过构建覆盖煤气、储能及自备电厂的全场景协同模型，本系统实现了从经验驱动的被动响应向算法驱动的主动预测模式的跨越。通过对电网峰平谷时段的精

准识别与能量搬移，企业外购电平均单价有效降低了5%至8%；通过对气柜压力的前瞻性调节，副产煤气的利用率显著提升，基本消除了非计划性放散，自发电量平均提升3%至5%。在运营维度上，自动化决策引擎将能源调度指令的生成周期从小时级缩短至分钟级，极大增强了系统对生产波动的抗干扰能力，为钢铁企业构建了基于量化计算的绿色低碳运行体系。

4.12.5 复制推广建议

本方案所沉淀的能源流转逻辑与多变量寻优算法，对于拥有复杂副产气体回收系统或自备微电网的工业场景具有较强的普适性。在钢铁行业内的跨产线迁移或向有色冶炼等类似场景推广时，需重点评估目标场景的数字化基础设施完备度。对于主推基于数学规划的优化模块，需确保 ERP、MES 以及能源管理系统能够提供高频且准确的物料平衡与实时工况数据，以支撑物理机理模型的有效运行。对于预测类模块，则需考察历史数据底座的完整性及其是否覆盖了足够的典型生产工况。建议优先选择电力成本占比高、具备一定储能调节能力且数据闭环相对完善的企业，通过明确技术前置条件来确保项目的快速交付与效益兑现。

4.13 焊接机器人控制参数优化技术方案与应用实践

4.13.1 业务背景介绍

4.13.1.1 业务背景描述

在工程制造领域，工艺参数调优是保障产品质量与生产效率的重要环节。以机器人焊接为例，焊接电流、电压、焊接速度等关键参数的设置，会直接影响焊缝宽度、焊缝高度以及焊接稳定性等质量指标。一旦参数设置不合理，就可能导致焊缝成形不良、强度不足甚至产品报废。在传统生产模式中，这类参数往往依赖经验进行调整。经验丰富的工程师通过反复试验，逐步找到一组可用的参数组合。然而，这种“人工调参”的方式存在明显局限：一方面，试错过程需要消耗大量时间与材料成本；另一方面，调参过程高度依赖个人经验，难以标准化和规模化复制。随着产品迭代速度不断加快，企业需要在更短周期内完成工艺参数调优，仅依靠人工经

验已难以满足实际需求。因此，如何通过智能化方法快速找到合适的工艺参数组合，成为企业提升研发效率和生产效率的重要课题。

4.13.1.2 业务痛点

在实际项目中，客户在开展机器人焊接参数优化时，普遍面临以下问题：

1) 系统复杂度高，难以建立精确模型：在焊接工艺中，系统由物理机理、材料特性以及经验规则共同作用形成。参数与结果之间的关系高度复杂，传统解析建模方法难以建立准确的数学模型。

2) 单次实验或仿真成本高：在焊接参数优化过程中，每一次实验可能涉及实机测试或高精度仿真，不仅耗时，而且需要消耗材料和设备资源。客户通常只能承受有限次数的试验，因此每一次实验都需要尽可能具有价值。

3) 参数维度高、组合空间巨大：实际系统中可调参数往往达到几十甚至上百个，参数之间还存在复杂的耦合关系。通过人工经验或简单的网格搜索，很难在可接受时间内找到高质量参数组合。

4) 实验结果存在随机性：在真实生产环境中，即使采用完全相同的参数组合，也可能因环境波动、设备状态变化或测量误差而产生不同结果，这使得优化过程必须具备一定的抗噪能力。

5) 优化目标与约束复杂：企业在进行参数优化时，往往不仅追求单一性能指标的最优，还需要同时兼顾生产成本、稳定性、安全性以及质量约束等多方面因素。

4.13.1.3 技术挑战

上述业务痛点使得传统优化方法难以直接应用，主要体现在以下几个技术挑战：

1) 无法获得系统解析模型或梯度信息；

2) 每次评估成本高，优化过程必须高效；

3) 高维参数空间导致搜索难度显著增加；

4) 实验结果存在噪声，需要具备鲁棒性；

5) 需要同时处理多目标和复杂约束。

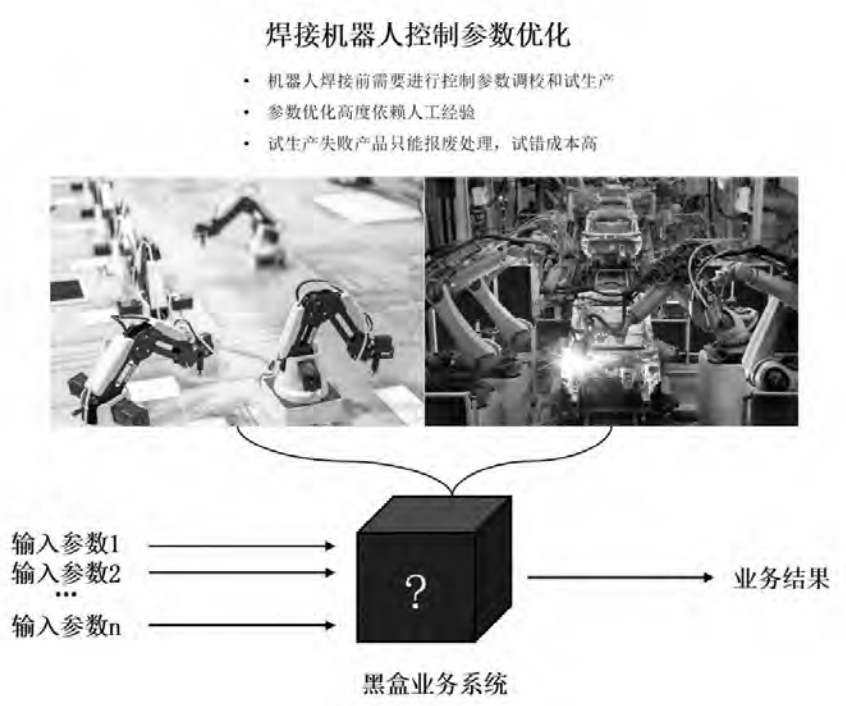

图4.36 焊接机器人控制参数优化场景示意图

在这种背景下，黑箱优化（Black-box Optimization）提供了一种可行的技术路径，如图4.36所示。黑箱优化通过仅依赖系统的“输入—输出”评估结果进行学习与决策，无需了解系统内部机理。算法通过不断尝试新的参数组合并利用历史实验结果进行建模与分析，逐步推荐更有潜力的参数方案，从而以较少的实验次数找到工程上可行的优质参数组合。

4.13.2 业务需求

4.13.2.1 客户整体功能需求

针对上述业务背景与痛点，参数优化系统需要具备以下核心功能：

1) 参数空间管理功能

支持定义可调参数的类型、取值范围及离散或连续属性，并构建参数搜索空间。

2) 实验评估接口

支持接入实际生产设备或仿真系统，对给定参数组合进行实验或仿真评估，并获取结果指标。

3) 智能优化算法模块

基于历史实验数据，通过黑箱优化算法自动推荐新的参数组合，以提高参数搜索效率。

4) 实验管理与结果分析

支持记录实验过程与结果，并对不同参数组合的效果进行统计与分析。

5) 可视化与决策支持

提供优化过程与结果的可视化展示，帮助工程人员理解参数变化对结果的影响。

4.13.2.2 集合和常量

在机器人焊接参数优化问题中，可以定义如下集合与常量：

1) 参数集合

$$P = \{p_1, p_2, \dots, p_n\}$$

其中 p_i 表示第 i 个可调参数，例如焊接电流、电压或焊接速度。

2) 参数取值空间

$$p_i \in [l_i, u_i]$$

其中 l_i 和 u_i 分别表示参数的下界与上界。

3) 评估指标集合

$$Y = \{y_1, y_2, \dots, y_m\}$$

其中 y_j 表示第 j 个评估指标，例如焊缝宽度、焊缝高度等指标。

4.13.2.3 业务目标

在机器人焊接参数优化场景中，系统的核心目标是通过智能优化方法，在有限的实验或仿真次数内，自动搜索并推荐一组性能优良的工艺参数组合。具体而言，需要在保证焊接质量稳定的前提下，综合优化多个关键指标，例如焊缝宽度、焊缝高度、焊接稳定性以及生产效率等。

此外，由于每一次实验或仿真都具有一定成本，系统还需要尽量减少实验次数，在有限试验预算内快速收敛到高质量参数组合，从而提高参数调优效率并缩短工艺开发周期。

4.13.2.4 业务约束

在实际应用中，参数优化过程需要满足多方面的业务约束条件，以确保推荐的参数方案在工程上可行且安全可靠。

首先，各类工艺参数通常具有明确的取值范围。例如焊接电流、电压以及焊接速度等参数均受到设备能力和工艺规范的限制，必须在允许范围内进行调整。

其次，优化结果需要满足质量标准要求。例如焊缝宽度、焊缝高度或焊接强度等指标必须达到既定质量规范，否则即使某些指标表现优异，该参数组合也不能被认为是可行方案。

此外，实际生产环境中还可能存在设备安全约束或工艺稳定性要求，例如避免电流过高导致设备损耗或焊接过程不稳定等问题。

最后，由于实验或仿真成本较高，客户通常会对优化过程设置实验预算约束，即优化算法需要在有限的试验次数内完成参数搜索，并尽可能获得高质量的优化结果。

4.13.3 技术方案

针对上述需求和挑战，我们提出了一套以贝叶斯优化（Bayesian Optimization，BO）为核心的黑箱参数优化解决方案，并结合不同类型的代理模型与采集策略，构建可扩展、可定制的参数寻优流程。其核心方法为贝叶斯优化框架，是一种面向高代价评估问题的序列式全局优化方法。

(一) 基本思想

使用已有的评估数据，构建一个概率代理模型来近似刻画黑箱函数；利用采集函数在“探索未知区域”和“利用已有最优解附近区域”之间进行权衡，逐步决定下一次评估的参数点。该方法特别适用于评估次数受限、函数形态复杂且缺乏梯度信息的优化场景。

(二) 解决方案整体流程

1) 问题定义与参数空间建模

明确优化目标（单目标或多目标）及必要的业务或工程约束；确定关键决策参数及其取值范围，支持连续、离散及类别型参数。

2) 代理模型构建

基于样本训练代理模型，常见选择包括：

a) 高斯过程（Gaussian Process，GP）

该模型适合连续参数、样本量较小的场景，可提供不确定估计。

b) 随机森林 / 梯度提升树

该模型在高维、离散或混合参数空间中更具鲁棒性。

c) 神经网络模型

该模型用于数据量较大或响应关系高度复杂的情形。

3) 智能迭代与主动采样

基于当前代理模型计算采集函数值，常用形式包括期望改进（Expected Improvement，EI）、上置信界（Upper Confidence Bound，UCB）与概率改进（Probability of Improvement，PI）。选择采集函数值最优的参数点进行下一轮实验或仿真；将新评估结果反馈至模型并完成更新。

4) 收敛判定与结果输出

当达到评估预算、迭代上限或性能改进趋于稳定时终止优化；输出推荐参数配置及对应的性能评估结果，为后续工程验证或业务决策提供依据。

整体解决方案架构如下图所示：

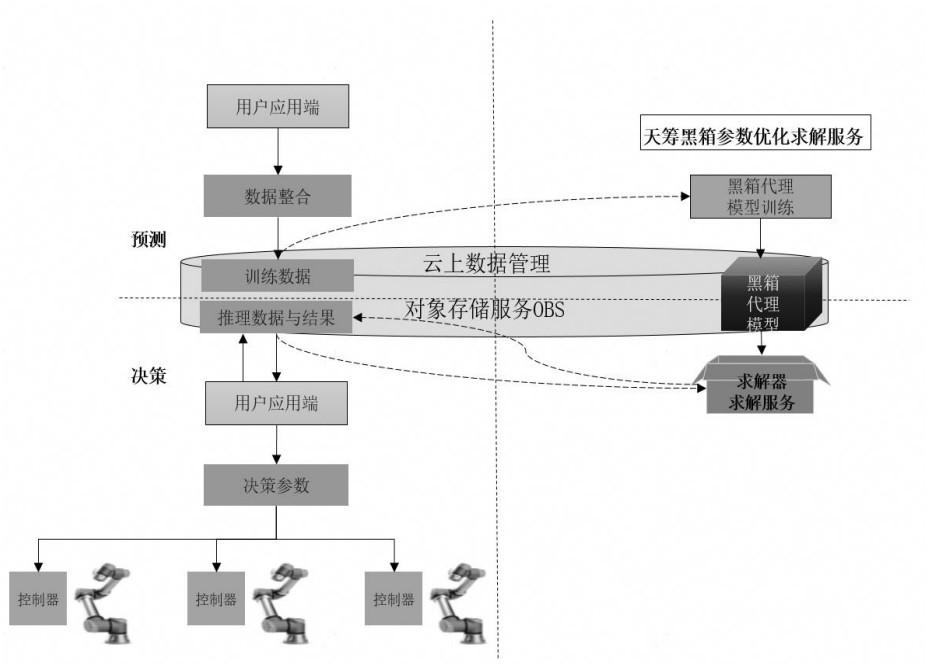

图 4.37 焊接参数优化解决方案

4.13.4 应用成效

依托在工程建模与智能优化领域的实践经验，天筹求解器优化团队将上述黑箱优化方法应用于多个真实工程与业务场景，如某焊接企业的机器人焊接工艺参数推荐场景等，并取得了稳定、可复现的应用效果。

1) 提高参数搜索效率，缩短优化周期

在涉及数十至上百个设计或工艺参数的问题中，通过序列化的智能采样策略，实现分钟级求解，相比传统试错或人工调参等方式，显著减少了必要的评估次数，加快了有效解的定位过程。

2) 增强对复杂非线性问题的适应能力

面对多峰、强非线性的响应面结构，优化过程能够在一定程度上避免过早陷入局部最优，逐步逼近工程可接受的高性能解或多目标权衡解。

3) 实现关键性能指标的可验证改进

在实际项目中，通过直接以最终性能指标为优化目标，协助客户在产品性能、能耗效率或质量稳定性等关键指标上获得可量化、可验证的提升，在相关测试集上实现了95%以上的推荐准确率。

4) 构建直观易用的交互体验

在项目落地过程中，通过设计简洁直观的用户界面，使客户能够轻松上手并高效完成参数优化任务，显著降低使用门槛。下图展示了用户输入焊件的相关信息，输入相关信息后即可进行黑箱优化。

图 4.38 用户参数配置示意图

总体而言，该方案通过将复杂系统“封装”为可评估的决策过程，为客户提供了一种工程上可落地、结果可解释、效果可验证的参数优化技术路径。

4.13.5 复制推广建议

1) 推广场景：该解决方案具备高度的通用性，不仅可应用于机器人焊接工艺参数的优化，还能够推广至其他需要进行复杂参数调优的工程与业务领域。例如，在制造业中，可拓展至材料配方优化、热处理工艺参数调整、化工反应条件优化等多种场景；

2) POC 测试：在数据收集完备（即每组数据均包含参数及其对应的评估指标）、各评估指标明确客观的前提下，该方案能够快速支持 POC，帮助客户在短时间内验证优化效果；

3) 落地路径：通过构建统一的参数优化平台，并将优化算法与具体业务系统或仿真环境进行对接，可实现该方法在不同应用场景中的快速复制与落地。这将显

著提升参数调优效率，降低试验成本，并减少对人工经验的依赖，为企业的智能化转型提供有力支撑。

4.14 新型铜合金材料设计技术方案与应用实践

4.14.1 业务背景介绍

4.14.1.1 业务背景描述

随着高端制造业的快速发展，新型高导电高强度铜合金已成为支撑电力传输、电子封装、轨道交通等多个关键领域稳定运行的核心基础材料，市场需求十分迫切。从具体业务场景来看，其应用覆盖生活与工业生产的重要环节：在电力传输领域，用于制作高压电缆、变压器等核心部件，需同时满足稳定输送大量电能、减少能源浪费，以及承受强风、冰雪等自然环境考验、不易变形损坏的要求；在电子封装领域，从 AI 芯片封装基板到新能源汽车连接器，均需其兼具优异导电性能与足够硬度，适配电子设备小型化、高集成化的发展趋势；在轨道交通领域，时速350公里以上高铁的接触线、电力金具等部件，依赖该合金实现高速运行下的稳定导电与耐磨耐用，为高铁提速升级提供支撑。基于此，相关研发项目聚焦该类铜合金，启动性能提升相关研究工作。

4.14.1.2 业务痛点

当前高端装备领域对新型高导电高强度铜合金的性能要求呈现“双高”趋势，即同时需要更高的硬度（以维氏硬度量化）和更优的导电性能（以导电率量化），但现有铜合金材料往往难以兼顾这两项核心性能，存在“硬度提升则导电率下降、导电率优化则硬度不足”的矛盾，无法充分满足电力、电子、轨道交通等领域的高端应用需求，成为制约相关高端装备升级发展的关键瓶颈。与此同时，传统新材料研发模式依赖反复试验迭代，存在研发周期长、试错成本高、仿真模拟难度大且成本居高不下的问题，而黑箱优化技术在该场景的应用尚未成熟，无法有效解决传统研发的痛点，导致研发效率难以提升，亟需通过专项研发结合黑箱优化技术，突破这一双重困境。

4.14.1.3 技术挑战

本研发项目的核心技术挑战包含两方面：一是如何实现铜合金材料硬度与导电性能的联合提升，破解两者之间的性能制衡难题，具体而言，需攻克“在提升铜合金维氏硬度、增强材料耐磨性和结构稳定性的同时，不降低其导电率，确保导电性能满足各领域核心部件的使用要求”这一关键技术难点；二是黑箱优化技术在该铜合金研发场景的适配与应用挑战，传统黑箱优化模型难以精准适配铜合金多变量、高耦合的性能优化需求，且存在样本数据获取难度大、模型训练成本高、优化精度与效率难以兼顾的问题，需攻克黑箱优化技术与铜合金研发场景的融合难题，实现两项核心性能的协同优化与研发效率的双重提升，使铜合金能够适配各高端场景的实际应用需求。

4.14.2 业务需求

4.14.2.1 客户整体功能需求

客户核心需求是解决铜合金研发过程中的性能瓶颈与研发效率问题，具体包括：破解铜合金硬度与导电能力难以兼顾的行业痛点，满足高端装备领域对材料“又硬又能导电”的核心使用需求；优化研发模式，解决传统试错式研发低效高耗、参数与约束条件复杂难把控的问题；应对材料性能机理复杂难以精准表述的难题，实现高效、精准的研发参数优化，推动铜合金研发成果产业化落地。

4.14.2.2 集合和常量

在新型铜合金材料设计问题中，可以定义如下集合与常量：

1) 参数集合

$$P = \{p_1, p_2, \dots, p_n\}$$

其中 p_i 表示第 i 个可调参数，包含元素组分参数、工艺开关与参数等。

2) 参数取值空间

$$p_i \in [l_i, u_i]$$

其中 l_i 和 u_i 分别表示参数的下界与上界。

3) 评估指标集合

$$Y = \{y_1, y_2\}$$

其中 y_1 表示铜合金硬度，y_2 表示铜合金导电能力。

4.14.2.3 业务目标

业务目标围绕性能、效率、精度展开，核心是打破传统工艺局限，实现铜合金硬度与导电能力的兼顾，满足高端装备领域对材料的高性能要求；同时替代传统经验试错式研发模式，缩短研发周期、降低人力物力投入成本以提升研发效率，解决参数与约束条件复杂难把控的问题，高效找到37项关键参数的最优组合、提升研发精度，突破行业长期存在的核心困境，推动铜合金研发成果产业化落地，提升产品市场竞争力。

4.14.2.4 业务约束

业务约束体现在多个方面，性能上，传统工艺中铜合金硬度与导电能力存在天然矛盾，提升一方必然导致另一方下降，难以兼顾，形成核心性能瓶颈；研发模式上，传统试错式研发周期长、成本高、参数覆盖不全面，而高精度材料性能仿真器耗时久、成本高，低精度预测模型精度不足，均难以高效支撑研发工作；参数与约束上，37项关键参数与20项约束条件相互影响，传统方法无法高效找到最优参数组合，制约研发效率与产业化落地；机理上，铜合金性能指标与元素组分、工艺参数之间的物理关系复杂，无清晰、准确的公式可描述核心机理，进一步增加了研发难度。

4.14.3 技术方案

面向这样一个，我们提出以贝叶斯优化为核心的方法搜索新材料参数，将高精度的仿真器与低精度的预测模型结合使用，应用多保真度寻优的方法在一定周期限制内获取更优参数。

(一) 核心方法：多保真度优化框架

我们希望结合较多次的低精度预测与较少次的高精度仿真的反馈评估结果，尽可能在较小的计算、时间成本下，获取相比只使用高精度仿真更好地结果。因此

利用天筹黑箱优化求解器中的多保真度优化特性，同时利用仿真与预测模型作为目标问题，遵循常规黑箱贝叶斯优化的流程对目标问题进行优化探索：

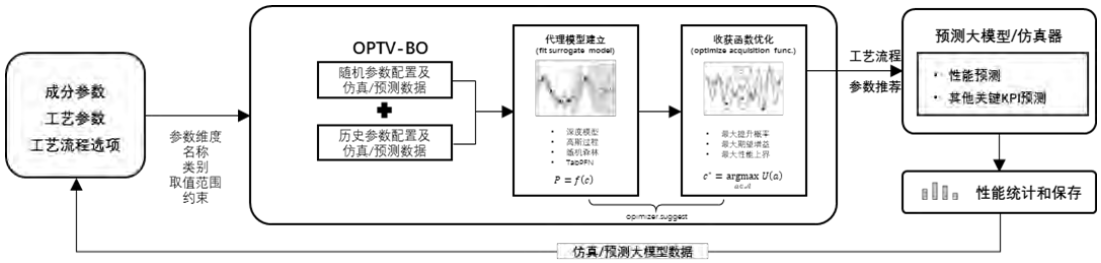

图4.39 黑箱优化技术应用于新材料研发的流程示意图

(二) 解决方案整体流程

步骤1：问题定义与参数空间建模。

这一步需要明确优化目标、参数空间、必要的业务或工程约束：

1) 目标一：维氏硬度（HV）：最大化提升，目标值≥750；

2) 目标二：导电率（%IACS）：最大化提升，目标值≥75%；

3) 参数空间：铜以及各类稀有元素的配置百分比，冷轧、热轧等相关工艺生产开关及细节控制参数；

4) 约束：元素组分加和为100%，铜元素占比大于某个阈值，稀有元素种类不超过4种，当对应工艺开关打开时，对应工艺细节控制参数才发挥作用等。

步骤2：代理模型构建。

基于离线样本训练代理模型并进行模型选型，本场景选择深度神经网络集成学习模型 Deep Ensemble 作为代理模型，以及期望改进（EI）作为采集函数。

步骤3：多保真度环境智能迭代与主动采样。

1) 配置多保真度优化方案，接入低保真度目标问题（对应低精度预测模型）和高保真度目标问题（对应高精度仿真器），选择合适的低保真度数据使用率（该场景为0.5）与高保真度环境评估频率占比（该场景为0.1），之后开始迭代优化；

2) 基于当前代理模型计算采集函数值；

3) 选择采集函数值最优的参数点进行下一轮评估，由优化器判断进行低精度预测或高精度仿真；

4) 将新评估结果反馈至模型并完成更新。

步骤4：收敛判定与结果输出。

1) 当达到评估预算、迭代上限或性能改进趋于稳定时终止优化；

2) 寻优结束后，从帕累托最优解集中，结合成本约束与工艺可行性，筛选出1组最优参数组合，验证优化效果，确保最优参数组合能够适配工业化生产需求。

4.14.4 应用成效

通过黑箱贝叶斯优化的新型铜合金材料设计方案，已经支撑相关头部企业客户完成新型材料的可行性验证与智能研发方案发布。实验验证中，针对高导电高硬度材料（期望导电率≥75%，维氏硬度≥ 750），基线多目标黑箱优化算法NSGA-III的结果达到导电率=67.2%，维氏硬度=622.1；天筹方案的结果在仿真环境上达到导电率=69.8%，维氏硬度=686.7，相比基线方案有明显提升，相比其他流行的优化算法也有明显优势。

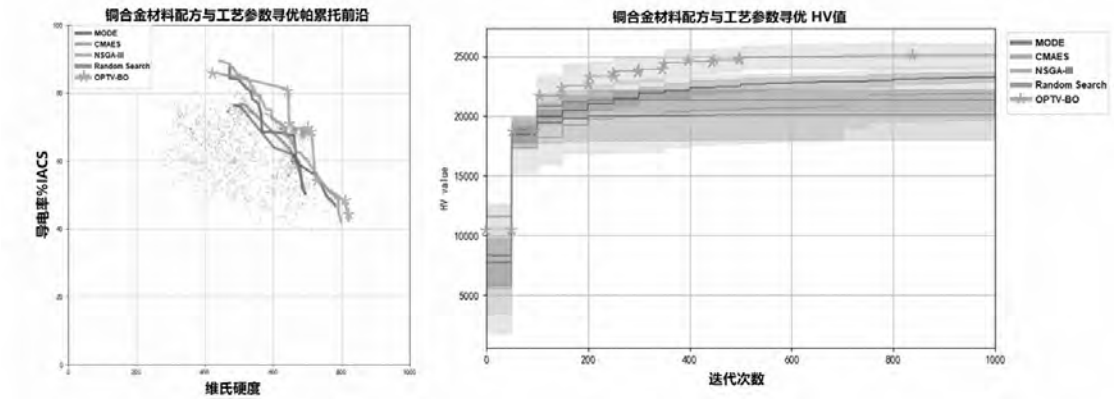

图4.40 铜合金性能多目标寻优算法对比示意图

4.14.5 复制推广建议

基于本案例中黑箱优化技术在铜合金研发中的实践经验，该算法方案可广泛推广至各类涉及高成本实验、高精度仿真的材料研发及科学研发领域，涵盖新能源材料、生物医药、航空航天等多个细分方向。此类研发场景在实际推进过程中，普遍存在实验/仿真单次投入大、耗时久，参数耦合关联复杂、难以精准把控，核心作用机理模糊且无法量化，以及传统人工经验试错模式效率低下、研发成本居高不下的共性特点，严重制约研发进度与技术突破。而黑箱优化技术可凭借其适配多参数、多约束黑箱问题的灵活特性，结合降本增效的核心优势，同时依托落地可靠、接入便捷的突出特点，有效破解各领域研发中的核心痛点，为各类研发工作提供高

效、精准的优化支撑，推动研发模式从传统经验驱动向智能算法驱动转型，助力各领域实现研发效率升级与关键技术突破，充分释放技术的规模化应用价值。

4.15 电力调度优化求解技术方案与应用实践

4.15.1 业务背景介绍

4.15.1.1 业务场景描述

电力调度是电网运行的核心问题，简单来说就是决定"什么时候开哪台发电机、发电功率是多少"，确保电力供需平衡。在传统电网中，调度员根据经验或简单规则安排机组启停即可满足需求。然而，随着风电、光伏等新能源大规模接入，以及电力市场化改革深入推进，电力系统变得日益复杂。

安全约束机组组合（Security-Constrained Unit Commitment，SCUC）就是在这样的背景下产生的核心决策问题。它的核心挑战在于：不仅要考虑发电机自身的启停逻辑，还要兼顾整个电网网络的输电安全。在新能源接入和电网规模化背景下，机组数量从数十台扩展至上千台，调度时段细致到小时级别，传统方法难以应对。

SCUC 的决策过程是将未来时间段（如24小时）划分为多个调度时段，每个时段用二进制变量表示机组启停状态，用连续变量表示出力功率。在数百台机组、数十个时段的组合下，决策变量规模可达数十万甚至数百万个。这些决策受最小启停时间、爬坡速率等设备限制，同时不同机组出力通过输电网络相互影响，需检查网络潮流是否越限。

SCUC 的决策变量分为两类：一类是表示每台机组在每个时段启停状态的二进制变量（0表示停机、1表示开机），另一类是表示每台机组在每个时段实际出力的连续变量。对于一个包含千台机组、调度周期为24小时的典型场景，决策变量总数可达数十万个，其中二进制变量和连续变量各占一半。这些变量的取值范围也受到严格限制，二进制变量仅限于0或1，连续出力变量则必须介于机组的最小技术出力和额定容量之间。

SCUC 问题的目标是最小化电力系统在未来调度周期内的总运行成本。总成本由多个组成部分构成，首先是各机组启停成本，即机组从停机状态切换到开机状

态所需的启动费用，包括燃料消耗、设备磨损和经济补贴等，其次是与出力水平直接相关的燃料成本，通常采用分段线性函数近似描述非线性燃料成本特性。在电力市场化背景下，目标函数还需要考虑碳交易成本、机会成本等经济性因素，这些成本项的合理建模对于找到具有实际经济价值的调度方案至关重要。

为保障电力系统的安全稳定运行，SCUC 问题受三类关键约束的共同制约。第一类是机组自身的技术约束，包括最小启停时间约束要求机组一旦开机或停机后必须持续运行或停运至少一定时间，避免机组频繁启停带来的设备损伤和经济损失，爬坡速率约束限制机组在相邻时段出力变化的最大梯度，确保机组能够安全平稳地跟踪负荷变化，出力上下限约束规定机组实际出力必须处于技术可行区间。第二类是系统平衡约束，要求任意时刻所有开机机组的总出力准确等于系统总负荷减去新能源出力后的净负荷需求，同时旋转备用容量需满足预先设定的安全裕度要求以应对突发情况。第三类是电网安全约束，基于直流潮流或交流潮流模型，对输电线路的传输功率和节点电压幅值进行严格校核，确保所有线路潮流介于其传输容量下限和上限之间，防止因潮流越限引发的线路过载或电压崩溃风险。电网安全约束使得调度决策不再仅仅是对各机组独立优化，而需要全网协同考虑机组出力分配对网络潮流分布的影响，这是 SCUC 问题区别于普通机组组合问题的核心特征。

4.15.1.2 业务痛点

1) 复杂度指数级增长挑战：SCUC 问题本质上是一个 NP-hard 的混合整数规划问题。在新型电力系统中，新能源机组大规模接入和调度时段扩展使得问题规模呈指数级增长。同时，新能源出力的不确定性进一步增加了问题随机性，需要额外应对场景覆盖和鲁棒性约束；

2) 时空强耦合约束难题：SCUC 问题呈现出显著的时空强耦合特性。在时间维度上，最小启停时间约束、爬坡速率约束等使相邻时段的决策存在强依赖关系，某个时段的调度决策需同时考虑其前后多个时段的可行性。在空间维度上，网络潮流约束将不同地区的机组决策紧密关联，某一区域机组的启停和出力调整可能触发远端输电线路的越限，需要全网协同优化。这种时空强耦合特性使得难以通过分解方法简化问题，同时导致割平面等下界提升方法的效果弱化；

3) 求解性能与精度双重要求：在实际电网调度业务中，电力调度方案需根据系统运行状态动态调整，大部分场景要求在半小时内完成一轮甚至多轮求解以保障时效性。传统求解器在千机组、百时段规模下的求解时间往往需要数小时甚至更长，难以满足实际生产需求。更为苛刻的是，在追求求解速度的同时，还需保证解的高精度。行业内以相对值间隙（GAP）作为衡量标准，GAP每降低0.1%，在电力市场的庞大体量下即可带来数亿元的经济收益。因此，SCUC问题求解器必须在时间窗限制内达到极低 GAP（如≤0.3%），在速度和精度双重制约下实现高效优化。

4.15.1.3 技术挑战

1) 要求在满足电网安全约束的条件下，算法求解时间尽可能快，要求在30分钟时间窗限制内达到极低 GAP（≤0.3%），在速度和精度双重制约下实现高效优化；

2) 要求性能超越国际商业求解器，同时确保求解成功率100%，不能出现求解失败或质量不达标的情况，确保算法在所有场景下具有良好的鲁棒性和一致性；

3) 要求实现软硬件全栈自研，求解器软件核心算法自主可控，满足电网调度重大核心场景自主可控要求。

4.15.2 业务需求

电网调度运行直接关系到能源供应安全、效率和经济性，因而对 SCUC 求解器提出多维度需求：求解能力方面，需具备处理千机组、百时段规模及强约束耦合问题的能力；求解质量方面，需在提升求解效率的同时保证解的精确性和可行性，避免因近似解导致的安全风险；易用性方面，需可平滑集成至现有调度系统，操作简便，降低部署和维护门槛。

4.15.3 技术方案

针对安全约束机组组合优化问题的特点，天筹求解器团队创新性地提出全方位技术改进方案，深度融合数学优化、AI 求解加速与高性能计算技术，构建了高效鲁棒的智能优化体系：

基于结构感知的预处理技术：深度挖掘 SCUC 问题的时空结构特征，通过隐含整数变量检测、分段线性函数斜率建模、非等价机组变量聚合等技术，系统化简化原始模型。该系列预处理技术可减少决策变量规模50%以上，从源头上降低问题求解难度，为后续算法加速奠定坚实基础；

智能启发式求解系统：针对 SCUC 问题找解困难的特点，设计了分层智能启发式框架。系统根据线性松弛解的整数不可行性程度动态选择求解策略，包括整数不可行条件控制、固定传播类启发式、局部搜索类启发式等。通过启发式与割平面紧密协同执行，实现下界提升与可行解搜索的良性循环，显著缩短求解时间并提升鲁棒性；

多策略并行求解框架：构建先进的多进程协同搜索机制，主进程运行基准策略，并行进程探索不同求解空间区域。该框架支持多种专家割平面、多种专家启发式的并行执行，有效利用现代计算硬件的并行能力，大幅提升大规模问题求解效率；

AI 辅助智能决策工具链：通过智能决策工具链对问题特征和求解过程进行智能化分析。工具链自动提取矩阵结构统计量，通过黑箱优化搜索最优参数组合，使求解器能够根据问题特征自适应调整参数，提升系统在不同工况下的鲁棒性和易用性。

图4.41 天筹求解器智能化电力市场求解HPC解决方案

4.15.4 应用成效

天筹 AI 求解器在第三届能源电子产业创新大赛-国产求解器技术专题赛中取得优异成绩，作为唯一成功求解全部10个赛题的队伍，充分验证了其技术方案的有效性和先进性，展现出优异的应用成效和行业影响力。

求解性能显著提升：通过系统性算法优化，天筹求解器在 SCUC 问题上实现跨越式性能突破。经过预处理技术、智能启发式系统、割平面协同优化等改进后，求解性能总体提升幅度接近90%，平均求解效率相比国际领先的商业求解器提升60%以上。

大赛成绩卓越：在速度赛道中，天筹求解器在5个测试模型上均达到收敛精度要求，在最大规模实例上仅使用规定时间1/3就达到了目标优度。在精度赛道中，5个模型 GAP 均控制在0.14%以内，展现了极高的求解质量。这些优异表现使天筹求解器成功夺得数学赛道第一名 (https://mp.weixin.qq.com/s/RRK4_3lqkaSEvfk5pZoSfQ)。

技术产业化价值突出：SCUC 问题求解效率的提升具备显著的产业应用价值。根据行业经验估算，在大型电网调度场景中，GAP 每降低0.1%即可产生数亿元的经济收益。天筹求解器的高性能、高稳定性特性能够支撑电网在新型电力系统背景下实现调度决策水平的整体提升，为保障电力供应稳定、高效运行提供核心算法支撑。

4.15.5 复制推广建议

基于本案例中天筹求解器在 SCUC 问题中的实践经验，该算法方案可广泛推广至各类涉及大规模混合整数优化决策的能源及工业领域，涵盖电力市场出清计算、电网检修计划优化、新能源功率预测与调度、能源互联网协同调度等细分方向。此类场景在实际推进过程中，普遍存在决策变量规模庞大(十万至百万级)、时空强耦合约束复杂难以分解、求解时效性与精度双重要求苛刻、以及传统商业求解器求解效率低下且成本高昂的共性特点，严重制约复杂系统的运行效率与经济效益。而智能优化求解技术可凭借其对混合整数规划问题的深度优化能力，结合结构感知预处理、智能启发式、并行求解等核心技术优势，同时依托高效求解、平滑集成、自主可控的突出特点，有效破解各领域优化决策中的核心痛点，为各类复杂优化问

题提供快速、精准的计算支撑，推动决策模式从依赖国外商业求解器向自主可控求解器转型，助力各领域实现运行效率升级与降本增效，充分释放技术的规模化应用价值。

4.16 云资源调度与碎片整理技术方案与应用实践

4.16.1 业务背景介绍

随着云计算产业迈向超大规模化与精细化运营，资源利用率优化已不仅是一种成本控制手段，更是云服务厂商核心竞争力的体现。随着虚拟化技术（Virtualization）的全面普及，云服务厂商面临着一个日益严峻的技术挑战：物理资源空间在动态分配过程中持续碎片化，导致数据中心内部有容量却无可用，直接制约了云服务厂商的售卖能力与经济效益。该现象通常被称为“资源碎片”（Resource Fragmentation）。

资源碎片的产生是虚拟机全生命周期管理中的多种动态因素交织的结果。物理服务器的硬件资源如 CPU、RAM、GPU、磁盘 I/O 以及网络带宽是有限且离散的。当用户请求创建不同规格的虚拟机时，底层调度器需要在满足各种物理与逻辑约束的前提下，为虚拟机寻找合适的宿主机（Host）。在数据中心资源充裕的初始阶段，调度器通常采用负载均衡策略以保证性能的稳定性。然而随着业务的运行，虚拟机的频繁创建与销毁会打破硬件资源在物理空间上的连续性。由于不同规格的虚拟机（如2U4G、8U16G、32U64G）具有不同的资源足迹（Footprint），且其生命周期（Lifetime）长短不一，导致物理节点上逐渐出现了一些零碎的、无法被任何标准规格利用的残留资源。

例如云集群中普遍存在“大作业”（需求资源量较大）与“小作业”（需求资源量较少）混合提交的场景，这种异构负载容易大量产生资源碎片：小作业由于规格灵活，往往会优先填充物理机中的剩余空间。当大量小作业散布在不同的物理节点上时，虽然从集群总量上看依然有足够的剩余资源，但由于没有单个节点具备足够的连续空间，仍会导致需要大规模资源的“大作业”因资源不足而陷入长时间的排队或调度失败。类似的现象在多维资源约束下尤为常见。例如，一个节点可能还剩余16核CPU，但内存仅剩2GB，但目前主流的大规格虚拟机要求16U32G，因此

这16核CPU很可能无法被利用，造成“资源搁浅”（Resource Stranding）。这种各资源维度之间的不平衡进一步加剧了资源碎片化程度

资源碎片化不仅仅是技术层面的效率问题，如下表所示，其负面影响可能广泛渗透到了企业的经营效率、运维质量及客户体验中。数据表明，大规模集群若缺乏有效治理，其因资源碎片化导致的资源浪费占比可达10%至25%，这对于云服务厂商而言是巨大的财务负担。因此，高效的资源碎片整理策略已成为云基础设施管理中的“必修课”。该问题可以被定义为多维装箱问题（Multi-dimensional Bin-Packing Problem）的变体，由于其具有高度的动态性与异构性，这一问题被证明属于强NP-难问题，目前在多项式时间内找不到最优解。

表4.19 资源碎片化问题的负面影响说明

目标	描述	负面影响
经济收益	因大规格实例无法创建导致的高价值订单流失	营收漏损、资产收益率（ROA）下降
能源效率	为了满足新需求不得不开启额外的空闲物理机，导致能耗激增	PUE 升高、运维成本增加
调度能力	集群响应时间（Response Time）变长，调度器搜索成本增加	SLA 风险、交付时延
性能干扰	碎片环境下的过度压缩部署（Over-packing）可能引发资源争抢	业务性能抖动、用户满意度下降
可持续性	频繁资源碎片整理（Defragmentation）增加系统不稳定性	基础设施加速折旧、有稳定性风险

4.16.2 客户需求

资源碎片整理的本质是通过对存量虚拟机的重新布局，将离散的空闲空间汇聚成连续的大块空间，以恢复集群的交付能力。在复杂的生产环境中，这一过程必须具备清晰的数学目标和严密的约束边界。

（一）业务目标

在云资源市场上，大规格实例（如高性能计算实例、大型数据库实例）往往具有更高的单价和利润空间。如果碎片整理仅追求整体利用率的提升，可能会导致空闲空间依然分散在各个节点，无法满足大规格实例的资源需求。因此受商业逻辑驱动，针对云资源售卖场景的碎片整理，其优化函数需要显式地衡量剩余空间的规格化程度，核心目标是最大化碎片整理后能够创建的指定规格虚拟机的数量。

（二）业务约束

1) 资源容量上限约束：每个物理节点上的 CPU、内存、网络、存储及特殊硬件（如 GPU）分配量之和不得超过其额定物理容量。对于 GPU 等非独占资源，还需考虑具体的切分与分配逻辑；

2) 故障域与亲和性约束：为保证高可用性，属于同一业务组的虚拟机通常被要求分散在不同的机架（Rack）或可用区（Availability Zone）。因此碎片整理过程中，虚拟机的移动不得违反原有的反亲和性规则；

3) 硬件制定约束：虚拟机可能依赖于特定的指令集（如 AVX-512）或硬件版本，因此整理后的新位置仍须具备与之匹配的硬件特性；

4) 迁移成本限制：热迁移需要短暂消耗资源以释放长期空间。为了避免大规模迁移任务拖累物理机性能，必须对单次执行的虚拟机总量及总内存进行限制，防止因负载骤增导致系统崩溃或网络拥塞，对业务产生副作用；

5) 服务等级协议（SLA）约束：迁移会导致虚拟机出现短暂的性能下降或毫秒级的停机。对于类似金融、电信等的敏感业务，需设定严格的迁移黑名单或时间窗口，确保 SLA 违约风险最小化；

6) 网络带宽阈值：算法必须确保迁移产生的流量不会挤占租户的正常业务流量，迁移任务的并行度受限于数据中心内部网络的预留带宽。

4.16.3 技术方案

资源碎片整理策略通常需要进行大规模局部调整，因此常用的一种求解算法是自适应大邻域搜索算法（Adaptive Large Neighborhood Search，ALNS）。这是一种基于“破坏-修复”机制的元启发式框架，其核心思想在于：通过对当前解的大规模破坏和智能重建，在极短时间内探索广泛的解空间，从而跳出局部最优解，逼近全局最优。

4.16.3.1 ALNS 的基本流程与自适应机制

ALNS在每一次迭代中都会根据历史表现动态选择破坏算子（d）和修复算子（r）。其伪代码逻辑如下：

Step 1 初始化：生成一个初始可行解S_{start}，并为所有破坏/修复算子赋予相等

的初始权重；

Step 2 算子选择：使用轮盘赌机制，根据当前权重w选择算子d和r。即权重越高，被选中的概率越大；

Step 3 迭代搜索：

- 应用破坏算子$d(S)$，移除一部分虚拟机进入“待重排池”。
- 应用修复算子$r(S)$，将池中的虚拟机按照优化目标重新插入。
- 根据接受准则（如模拟退火）决定是否接受新生成的解。

Step 4 权重更新：在一个迭代周期结束时，判断是否达到终止条件，若没有则根据算子在这一周期内发现更优解的效果，调整其权重得分，之后转Step 2。

4.16.3.2 破坏算子设计

破坏算子的设计目的是创造可行解的空间，空间太大则难以进行有效搜索，空间太小又容易偏离全局最优，因此其策略多样性决定了搜索的深度和广度。可能的破坏算子设计包括：

1) 随机移除：随机选择一定比例（如 5%-15%）的虚拟机进行迁移。该算子有助于算法保持探索性（Exploration），防止陷入特定的局部最优解；

2) 高碎片化移除：计算物理机上资源的“搁浅”程度，移除处于高碎片化状态主机的虚拟机。例如，某主机 CPU 占用 90%但内存仅占用 10%，则移除该主机上的部分虚拟机，以尝试释放被封锁的内存空间；

3) 高关联性移除：假设相似的虚拟机放在一起进行重排更容易达成规整的布局，可以根据租户属性、应用类型或资源规格相似度进行移除；

4) 高空闲度移除：以腾出完整主机为目标，优先移除已经接近空闲的主机上的剩余虚拟机，从而将该节点完全清空，以承载超大规格实例。

4.16.3.3 修复算子设计

修复算子的任务是往空闲主机中重新插入“待重排池”中的虚拟机，其策略目标应最大化可售卖的规格及数量，以下列出了几种常见的修复算子：

1) 价格贪心插入：评估每个候选位置对大规格资源完整性的影响，优先选择那些不切断连续空间的位置进行插入，以规避对核心空闲空间的影响，确保剩余资源的售卖价值最大化；

2) 遗憾值插入：计算将虚拟机放入第一优选位置与第二优选位置之间的目标价值差（即遗憾值）。优先插入遗憾值大的虚拟机，以防止安置选择极少的虚拟机在后期无处安放；

3) 紧凑装箱插入：借鉴 Best-fit Decreasing 思想，将虚拟机优先塞入那些已经较满的主机，从而尽量保护未分配主机的资源连续性；

4) 维度平衡插入：优先选择能使节点的 CPU/内存等维度利用率趋向平衡的位置插入，减少各维度资源不平衡导致的资源搁浅。

4.16.3.4 基于大模型自动算法设计

可以针对 ALNS 算法框架中最核心的“破坏”算子和“修复”算子，使用基于大模型的自动算法设计技术自动设计新算子，并且在数据集上进行评估得到算子的性能，配合演化框架逐步迭代，得到更优的“破坏”和“修复”算子设计。

整体的算法流程如下图所示：

1) 大模型生成若干新算子（可参考已有算法，也可完全重新生成）；

2) 将新算子嵌入 ALNS 框架，得到若干新版本 ALNS 算法；

3) 对这些新版本 ALNS 算法在指定数据集上进行评估；

4) 保留评估价值较大版本的算法，淘汰较差算法，构成下一代算子种群的参考算法；

5) 重复前面 4 个步骤，直到算法收敛或者达到最大迭代次数。

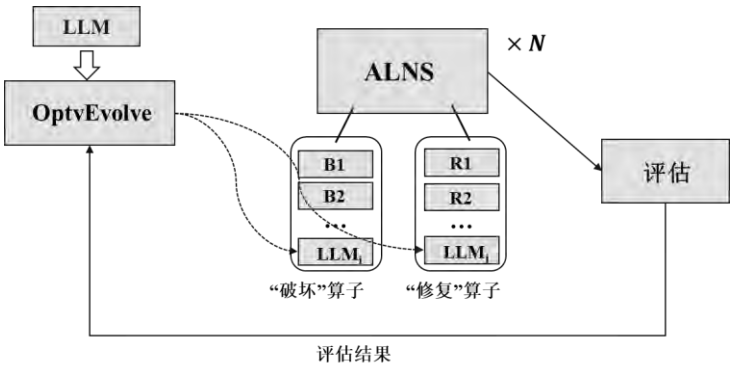

图4.42 基于大模型的ALNS框架算法技术架构

4.16.4 应用效果

基于真实场景抽取的数据验证，通过OptvEvolve框架对关键算子进行自动设计构建的ALNS框架在资源碎片整理任务中取得了显著的效果，主要体现在：

1) 得到的新版本算法在所有测试用例中均不弱于原始版本的算法，体现出基于大模型自动算法设计的解决方案的鲁棒性；

2) 在部分用例中改进幅度超过 15%，体现出基于大模型自动算法设计的求解框架能有效寻找更高效的算法方案；

3) 所有新生成的算子均包含详细的算法思路，整体具备较高的可解释性。

4.16.5 复制推广建议

1) 突破领域壁垒，具备广谱适配性

本解决方案依托大语言模型（LLM）驱动的自演化算法发现技术，其底层逻辑具有高度的通用性与可迁移性。除已验证的云资源调度场景外，本方案可广泛横向拓展至智慧物流、金融量化交易、工业排产、电力调度等任何具备“既有基线算法”且“评价指标明确”的复杂决策场景。

2) 重塑研发范式，实现算法工业化生产

通过构建中心化的自动算法演进平台并集成大规模算力集群，将传统的“人工经验试错”模式重塑为“机器自闭环优化”模式。这种模式不仅能指数级缩短算法迭代周期，更能突破人类思维定式，探索出高性能、高鲁棒性且具备原生逻辑可解释性的新型算子，真正实现核心算法的批量化、标准化产出。

3) 沉淀技术资产，构建自进化智能化底座

方案的推广不仅是工具的复制，更是算法能力的沉淀。随着接入场景的增多，平台将形成从“经验提取”到“自动评估”的良性循环，助力企业构建自进化的算法基础设施，在多变的业务需求下保持算法竞争力的持续领先。

4.17 汽车外流场大规模分布式数值仿真技术方案与应用实践

4.17.1 业务背景介绍

4.17.1.1 业务场景描述

流体仿真应用在汽车行业的概要设计、系统设计与研发的重要环节，指导设计研发，缩短研发周期，降低研发成本。汽车厂商通过汽车外流场计算，改进车身及底板设计，减少整车阻力。传统的车身外形设计主要依赖于风洞试验，通过分析车身表面的风阻情况来指导、优化车身设计。然而，这类方法存在成本高、设计流程慢等问题。上世纪七八十年代，随着计算流体力学的快速发展，高精度格式、湍流模型的提出、高性能计算机的快速发展，越来越多的汽车智造厂商转向通过高精度数值模拟的方式代替大部分风洞试验（接近60%），从而降低成本、缩短研发周期，为行业创造巨大的市场价值[159]。

4.17.1.2 业务痛点

1) 仿真精确模拟难：无法直接应用直接数值模拟，同时车辆外形极其复杂且容易产生强剪切层、流动分离、尾涡和近壁湍流，对网格分辨率、数值耗散和边界处理极度敏感，导致高精度结果难以收敛[160]；

2) 网格引入数值负担重：复杂几何、自适应网格，会导致矩阵性质变差、通信模式复杂[160]，直接拉高线性求解器的难度和成本，成为端到端仿真的核心性能瓶颈；

3) 线性求解占比时间多：在高分辨、大规模外流场仿真中，稀疏线性方程组求解往往占到总耗时的 50% 以上；当网格量达到数十亿级（20 亿规模）时，传统迭代法（CG/AMG 等）在分布式集群上要同时做到“准、快、稳”依旧充满挑战；

4) 线性求解并行效率低: 如何在矩阵求解部分缓解通信-计算耦合、在非均匀网格上实现负载均衡，依然是限制大规模并行效率的关键难题。

4.17.1.3 技术挑战

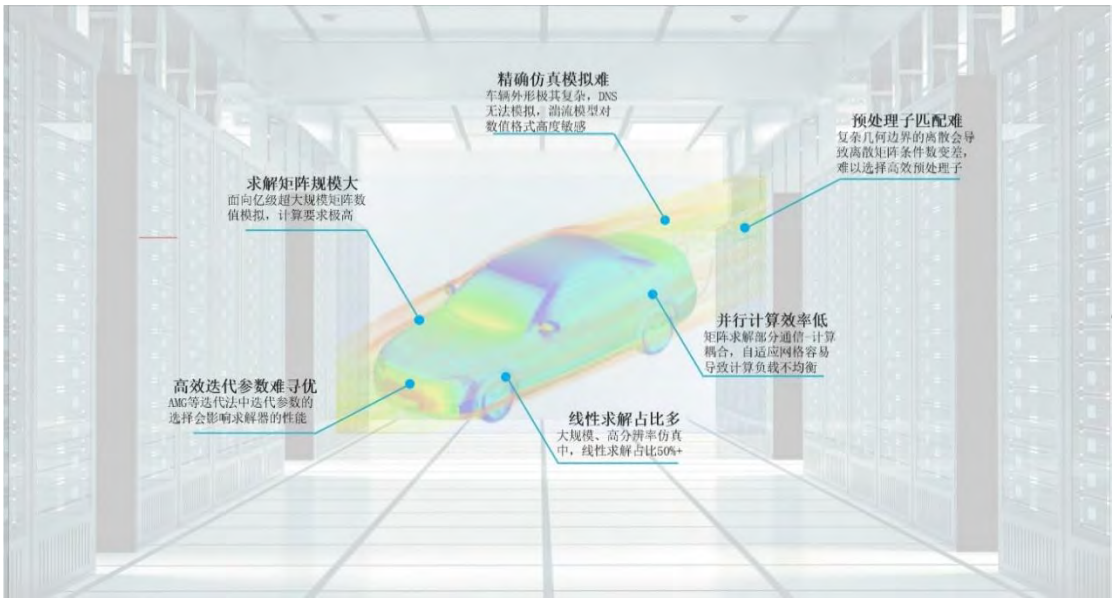

图4.43 汽车外流场计算数值挑战示意图

如图4.43 所示，该图展示了大规模汽车外流场仿真中存在的计算挑战，这其中可以概括为以下几点：

1) 精确仿真模拟难：由于车辆的外形复杂，模拟所需的网格数量难以直接应用直接数值模拟（DNS）；湍流模型又对数值格式高度敏感；

2) 求解矩阵规模大：利用万核资源，面向十亿级网格超大规模数值模拟，对于数值稳定性、计算效率以及并行性都提出了巨大挑战；

3) 并行可扩展性差：传统算法的强扩展性随着核数的增加会变差，大规模并行中的通信瓶颈与同步开销对数值求解有较大影响，从而导致计算性能下降；

4) 高效参数难寻优：为高效求解超大规模仿真，对迭代法中的参数选择提出了更高的要求；

5) 预处理子匹配难：在复杂边界条件下，要保证尽可能少迭代次数且保证收敛精度，需要给出更高效的预处理子，尽可能保持在仿真工况下保持高效求解效率；

6) 线性求解占比多：此类大规模、高分辨率的仿真中，线性求解器的耗时占比超过 50%。

4.17.2 业务需求

客户用雷诺平均RANS模型，湍流模型用$k-\varepsilon$模型[161]，在万核规模上计算网格数量超20亿规模的汽车外流场仿真问题。物理模型经过方程组离散之后，其中耗时占比最高的部分（超过 60%）就是离散方程组求解。客户要求天筹NCS数值计算求解器在端到端求解性能上能超越竞品50%以上。同时在并行效率上天筹数值计算求解器应保持竞品的领先性。

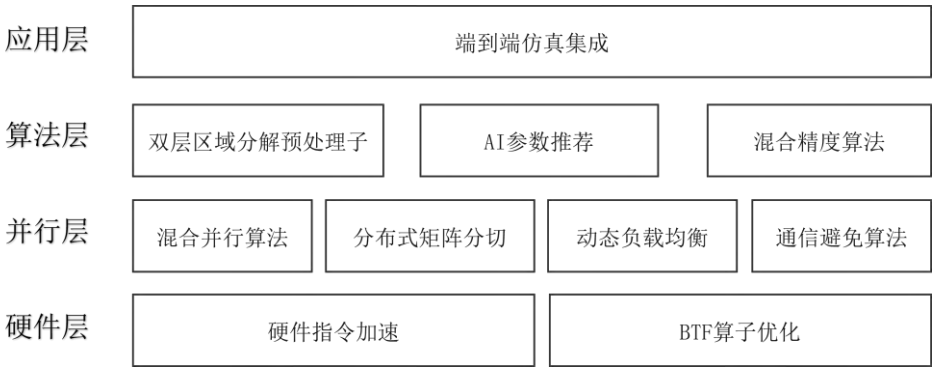

图4.44 天筹数值计算求解器技术架构

4.17.3 技术方案

4.17.3.1 技术架构设计

天筹数值计算求解器为了应对实际需求，在硬件层、并行算法、数值算法以及端到端仿真各个层级都进行了全面优化，从而在大规模应用场景中表现出色。基于底层BTF加速技术和硬件指令加速的基础，天筹NCS数值计算求解器在并行层采用了混合并行、分布式矩阵分切、动态负载均衡和通信避免算法等高效并行策略。在数值算法层，该求解器利用了AI技术、高效预处理技术和混合精度算法，使其在端到端仿真获得显著性能提升。

4.17.3.2 关键技术模块

1) 底层算子多级加速优化：优化 BTF 算子层（SpMV 访存/向量化），并加速硬件指令集，使得其在稀疏和稠密矩阵向量乘中获取最佳性能；

2) 通信避免算法（Pipeline 技术）：通过将规约压缩为一次非阻塞规约并与 SpMV 及预条件子计算重叠，减少显式等待的通信开销，同时提供异步范数计算和异步 DOT 计算等基础算子，提升整体并行效率与稳定性；

3) 混合精度迭代法：在矩阵向量乘阶段用低精度，再迭代中双精度进行误差矫正，实现“低精度加速 + 高精度校正”的一体化技术方案，显著提升大规模线性方程组求解的端到端效率；

4) 领域特化预处理技术：结合细层的 Additive Schwarz Method 和粗层的 Schur 补全局粗网格算子，通过减少内存冗余和加速 Setup，显著改善大规模耦合线性方程组的求解效率和可扩展性；

5) AI 推荐迭代法参数：通过特征提取和黑箱优化生成最优参数组，并训练代理模型自动推荐参数，实现自适应调参，提升求解器在通用场景下的鲁棒性和易用性，同时支持客户侧增量训练以进一步优化性能。

4.17.4 应用效果

4.17.4.1 计算效率

对于压力场和速度场求解，天筹NCS求解器与PETSc求解器的参数选择如表4.20所示。

天筹NCS求解器和PETSc求解器的端到端求解效率数据如表4.21所示，不同计算核数条件下的计算时间对比如图4.45所示。由表4.21和图4.45所示结果可知，天筹NCS求解器计算的计算耗时显著少于PETSc求解器，相同核数下计算效率提升可达50%。

表4.20 天筹NCS与PETSc参数选择表说明

求解器	**PETSc求解器**		**天筹NCS**	
物理量	压力	速度	压力	速度
KSP方法	pipeline cg	bicgstab	pipeline cg	bicgstab
预处理器	bjacobi		bjacobi	
收敛残差	1e-7	1e-7	1e-7	1e-7
相对残差	0.01	0.1	0.01	0.1
迭代次数	1000			

表4.21 端到端求解效率及并行效率说明

进程配置 节点数 × 核数	NCS求解器		PETSc求解器求解器		效率对比 NCS vs PETSc 求解器
	时间/s	并行效率	时间/s	并行效率	
100×10	5993	100%	9088	100%	152%
100×20	4896	61.2%	6230	72.9%	127%
100×40	4105	36.4%	6098	37.2%	149%
200×40	2790	26.8%	3919	28.9%	140%
200×50	2295	26.1%	3347	27.1%	145%

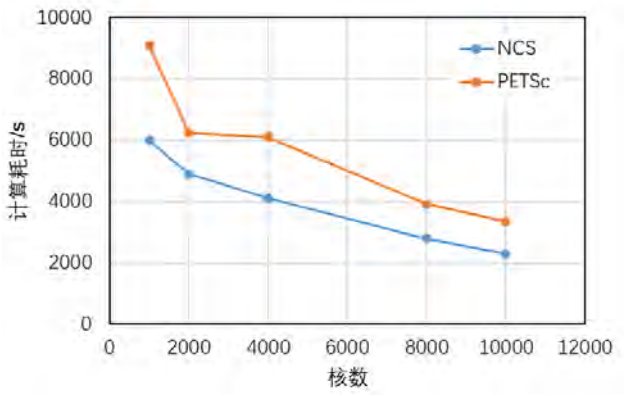

图4.45 天筹NCS和PETSc计算耗时对比示意图

求解效率对比采用公式：

$$\xi = \frac{T_{竞品}}{T_{NCS}} * 100\%$$

并行效率计算公式：

$$\eta = \frac{n_0\, T_0}{nT} * 100\%$$

其中n和T对应计算采用的核数和计算时间，n_0和T_0对应基准进程配置（节点 × 核数/节点）下的核数和计算时间。

4.17.4.2 并行效率

在并行效率方面，如表4.22和图4.46所示，采用天筹NCS求解器的效果和PETSc求解器表现接近，4000核以下并行效率随着核数的增加而降低，超过4000核后趋于

稳定（约30%）。进一步考察稀疏矩阵求解时间及并行效率，稀疏矩阵求解的并行效率随核数的变化规律与端到端求解相同，超过4000核后的并行效率的稳定值略高于端到端求解。

表4.22 稀疏矩阵求解时间及并行效率说明

进程配置 **节点 × 核**	**NCS求解器**	
	时间/s	**并行效率**
100 × 10	5425	100%
100 × 20	4406	61.5%
100 × 40	3567	38.0%
200 × 40	2115	32.0%
200 × 50	1725	31.4%

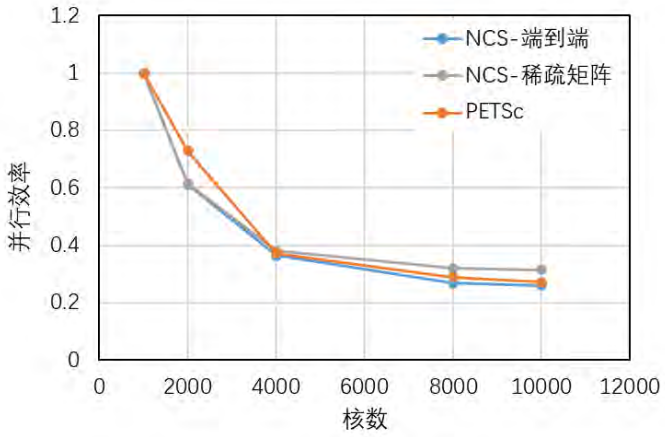

图4.46 天筹NCS和PETSc求解器并行效率对比示意图

4.17.4.3 仿真结论

基于实验方案数据可得，该案例在汽车外流场场景中采用 NCS求解器后，相同核数下的计算效率显著优于PETSc求解器，综合性能提升最高可达50%。在并行表现方面，天筹NCS与PETSc求解器表现整体接近：4000核以下并行效率随核数增加逐步下降，超过4000核后趋于稳定，体现出大规模并行进入稳定扩展区间。得益于通信避免算法、双层预处理子以及混合精度优化等关键技术，天筹NCS求解器实现了明显的工程化效果：具备万核级扩展能力，在典型CFD场景中计算效率优于竞

品，并已完成十亿网格规模端到端仿真验证，已具备行业落地基础能力。面向未来，方案将进一步推进两条主线：一是完成昇腾异构加速模块开发，目标在现有基础上再提升约20%性能；二是推动 AI4Solver在结构力学场景的多目标参数优化落地，例如实现自适应收敛阈值调整等智能化能力。

4.17.5 复制推广建议

本白皮书剖析了高精度CFD技术在汽车外流场优化中的核心应用范式。通过构建全尺寸整车气动模型，本案例精准捕捉了车身表面的复杂分离流、尾涡结构及气动干扰效应。分析结果不仅量化了风阻系数（Cd值）的细微演变，更直观揭示了气流对车辆升力、侧向稳定性及风噪特性的影响机制。

相较于传统风洞试验，本白皮书所展示的方案显著缩短了研发周期，降低了物理样机成本，为造型设计与工程性能的平衡提供了坚实的数据支撑。无论是优化后视镜细节以减少盲区涡流，还是调整底盘平整度以提升高速续航，该仿真流程均能提供可落地的改进策略。本白皮书旨在为主机厂研发部门、工程设计公司及科研机构提供验证数字化孪生能力、提升整车能效表现的标准化范本，助力下一代低风阻车型的高效开发。除此外，本白皮书所述的高精度仿真方法与自动化工作流具备高度通用性，可无缝迁移至以下领域的批量仿真场景：

1) EDA 电子散热仿真：针对芯片封装或 PCB 板设计，可批量复用热仿真流程。在固定冷却架构下，快速遍历数百种元器件布局方案或不同功耗负载工况，自动生成结温分布云图与热点预警报告，加速高性能计算芯片的热设计迭代；

2) 消费电子电磁兼容仿真：在智能手机或可穿戴设备开发中，利用参数化扫描技术，批量评估不同天线位置、外壳材质及内部堆叠对电磁辐射的影响。可一次性生成多频率下的远场辐射方向图与 S 参数曲线，快速筛选出符合电磁兼容标准的最佳设计方案；

3) 新能源电池包热失控扩散仿真：面向储能系统或动力电池包，建立标准化的热滥用触发模型。批量模拟不同电芯排列方式、隔热材料厚度及冷却流道设计下的热蔓延路径，快速输出安全窗口数据，为电池系统的安全策略制定提供规模化数据支撑；

4) 航空航天翼型族气动性能评估：针对无人机或飞行器机翼设计，建立参数化翼型库。批量计算不同攻角、马赫数下的升阻特性，自动生成气动极曲线数据库，辅助设计师在广阔的设计空间内快速锁定最优翼型构型，大幅缩短飞行器总体设计周期。

5 求解器技术未来发展方向和展望

5.1 数学优化求解技术发展方向和展望

数学优化作为现代科学与工程决策的核心方法论，已从早期的线性规划与凸优化理论，发展为涵盖大规模非凸优化、混合整数规划、不确定性优化以及智能优化的综合技术体系。在人工智能、能源系统、智能制造、交通网络和金融工程等领域的推动下，优化问题的规模、复杂性与实时性要求持续提升，促使算法、算力与建模方式发生深刻变革。

本白皮书将从大规模优化、非凸优化、AI融合等角度分析数学优化技术未来发展方向，并针对数学优化求解器技术的发展提出展望和行动建议。

5.1.1 数学优化求解技术发展方向

（一）大规模优化

随着深度学习与大规模数据分析的兴起，优化问题的变量规模已从百万级别扩展至十亿级别。随机梯度及其变种在工业系统中得到广泛应用，分布式优化框架成为基础设施的一部分。

当前大规模优化的核心瓶颈已从“单次迭代复杂度”转移到“通信与系统协同效率”。在分布式环境中，节点间通信开销往往超过计算开销，梯度同步成为制约收敛效率的重要因素。因此，梯度压缩、低精度计算与异构硬件协同计算成为重要研究方向。

（二）非凸优化与全局优化

现实世界的优化问题大多具有非凸性，尤其在深度神经网络训练和组合优化领域。尽管非凸问题理论复杂，但实践表明，大规模非凸优化往往存在“良性结构”，例如鞍点易于逃逸、局部极小点质量较高等现象。

本白皮书已介绍了包括LP、MILP、MIQP等多个凸数学规划问题（指问题的连续松弛是凸优化问题）的高效求解算法。然而，数学规划中的许多非凸问题，例如

混合整数非线性规划（Mixed Integer Nonlinear Programming，MINLP），在搜索全局最优解时仍面临规模受限与收敛缓慢问题。

当前趋势正在向“学习辅助优化”转变。强化学习与图神经网络被用于预测分支定界策略与变量选择顺序，从而显著降低搜索空间。未来十年，人工智能可能成为组合优化中的加速模块，实现启发式与精确算法的深度融合。

（三）优化与人工智能的融合

优化与人工智能的融合正在形成新的技术范式。一方面，深度学习模型的训练本质上是大规模非凸优化问题；另一方面，机器学习技术也被用于改进优化算法本身。近年来，研究者开始探索算法自动设计与自动调参。通过机器学习或大模型对传统数学优化算法实现或算法调度策略进行改造和持续演化，生成效率更高的新算法。此外，一些学者也在尝试通过基于大模型的数据生成框架[162]，以及新的注意力机制设计进行更深层次的学习[163]，来解决AI在组合优化问题上泛化性不足的问题。

（四）数学规划与约束规划技术融合

基于松弛问题求解和分支定界技术的数学规划求解路径，以及基于冲突驱动的子句学习（Conflict-Driven Clause Learning，CDCL）的约束规划求解路径，可以通过合适的组合形成更有力的离散优化问题解决方案。

在算法特点上，数学规划擅长利用线性或凸松弛技术求解包含连续变量的大规模问题。约束规划则擅长通过逻辑约束和区域缩紧技术快速搜索调度、序列和多种组合优化问题。在解决实际应用问题时，两者的技术可以相互融合实现算法加速，例如：

1) 在数学规划内调用约束规划技术：约束规划可以求解数学规划求解过程中产生的特殊子问题，例如路径优化问题。此外，约束规划也可以增强数学规划的原始启发式、域传播、冲突分析等模块的性能。

2) 在约束规划内调用数学规划技术：数学规划可以通过松弛问题求解，例如LP松弛求解，得到当前约束规划搜索树的下界评估。

（五）异构计算与分布式计算

计算架构的革新是数学优化求解技术近年来的新风向。经典数学优化求解算法通常基于CPU计算实现。随着问题规模的扩张，即使是存在多项式时间算法的线

性规划问题也提出了新的挑战问题。经典的CPU单纯形法和内点法执行效率及内存开销已无法支撑亿级变量模型的求解。于是，一阶方法结合GPU并行计算的新模式正受到越来越多研究者的关注。此外，部分组合优化问题的启发式算法也出现了适配GPU并行计算的版本。

分布式并行是分支定界算法的重要加速技术。传统的分布式架构虽然能带来一定的加速，但通信开销的引入使得分布式资源扩张时无法有效提升并行加速比。结合最新的超节点技术，超低通信损失的新架构可能会对基于分支定界算法的混合整数规划等模型求解带来重大革新。

5.1.2 数学优化求解技术展望

数学优化求解技术正站在“算法智能”与“算力革命”交汇的历史节点。未来十年，求解器将从单一计算工具演进为融合AI、异构、边缘计算的智能决策基础设施。具体来说：

6) AI与求解器深度融合：一方面，AI可以使能数学优化求解器，进行策略选择和算法进化；另一方面，数学优化求解器也可以使能AI，加速大模型推理和改善硬件资源调度。随着大模型技术的发展，新型AI将更显著地促进数学优化求解技术的革新。

7) 异构计算：以连续优化一阶算法和离散优化启发式算法为起点，GPU改造和加速传统算法将覆盖更多领域。异构计算正在突破传统CPU计算设下的优化模型规模上限。

8) 分布式计算：随着线性规划和整数规划技术的日趋成熟，越来越多的非线性甚至是非凸数学规划问题成为学术和商业求解器重点突破的领域。基于新的超节点分布式架构，这些困难问题的求解效率将在算法和算力共同发展的趋势下得以显著提升。

5.2 数值计算求解技术发展方向和展望

数值计算求解器是现代科学计算与工程应用中的核心工具，广泛应用于航空航天、气象预报、药物设计、金融分析等关键领域。随着计算需求的日益增长和算

力的不断提升，求解器技术正在加速向高效、智能、通用和自主的方向发展。本文从算法与硬件适配、求解器本身技术优化、未来发展方向、下一代求解器的形态与解决方案以及提升性能竞争力等方面，系统分析数值计算求解技术的前景，提出未来技术的演进路径。

5.2.1 数值计算求解技术发展方向

（一）算法与硬件高度适配化

未来算法不仅是追求算法的优劣，更重要的是围绕硬件进行高度定制化和适配化设计。现在算力的增长远快于计算过程中数据的搬运能力，很多稀疏矩阵线性代数操作仍然是memory-bound。因此，带宽、访存模式、通信延迟等将会是限制数值计算求解器效率的重要难题。未来硬件与算法适配的新特性会在与稀疏格式与硬件相适配、数据布局与硬件相融合、按内存层级来设计算法等关键技术上。

（二）算法优化与高度并行化

数值计算求解器的性能直接依赖于求解算法的优化。在未来，求解器将继续优化传统的求解方法，特别是在处理稀疏矩阵、非线性方程和特征值问题时，算法的效率与精度将成为重要研究方向。例如，对于大规模稀疏矩阵，传统的直接法如LU分解和Cholesky分解将在并行化的基础上进行优化，以应对大规模计算需求。而对于非线性方程组求解，未来将结合机器学习优化迭代初值和路径选择，从而提升收敛速度与求解精度。

在大规模并行计算的趋势下，万核和万卡计算平台的出现和普及，使得求解器的性能得到了质的飞跃。结合现代并行计算架构和AMG等具备高可扩展性的算法，可以显著提高求解效率和精度。

（三）异构计算与云计算支持

计算资源的多样性使得异构计算成为求解器技术发展的一个重要方向。未来的求解器将不仅仅依赖于传统的CPU，还将支持GPU、NPU和量子计算等多种计算平台的协同工作。随着异构计算架构的推广，求解器将在处理超大规模、复杂问题时表现出更高的计算效率。此外，云计算和基于Web的求解平台将使求解器能够支持大规模分布式计算，用户无需依赖本地计算资源，便可通过云端平台进行高效求解。

（四）**智能化求解器与自适应算法**

随着计算需求的多样化和复杂化，求解器将逐步实现智能化、自适应化。自适应网格技术和智能优化方法的结合将使得求解器能够根据具体问题自动选择最优的解法，降低计算成本并提高求解精度。未来，机器学习将在求解器中起到越来越重要的作用，帮助自动调节算法参数、选择最优预处理方法，并动态调整求解路径。

（五）**算法演进自动化与求解器平台化**

尽管当前广泛使用的数值计算求解器算法很稳定，但算法是陈旧的且没有应用最新的编译特性。未来数值计算求解器中的算法将会根据矩阵特征、硬件环境与运行状态的改变进行自动演化到最佳求解状态。与此同时，求解器也将会从分散的算法实现走向平台化发展，形成统一的接口、组件化架构、异构支持以及自动调优能力兼备的求解平台，从而极大提升其可扩展性、可移植性以及工程化能力。

5.2.2 数值计算求解技术展望

数值计算求解器将呈现出平台化、生态化、定制化的趋势。求解器将不再是单一的数学工具，而是一个高度集成的解决方案平台，涵盖建模、求解、分析、可视化等多项功能。通过云端架构和微服务化设计，用户可随时随地进行高效计算，而不必依赖本地计算资源。此外，求解器将具备跨学科融合能力，支持不同领域的定制化解决方案。下一代求解器将通过智能化和自动化的提升，减少人工干预的需求，优化计算流程，提升计算效率和精度。机器学习的嵌入将使求解器能够根据问题特点智能选择最合适的算法和策略，从而实现“即用即得”的自动化求解服务。这种产品形态将极大降低用户的使用门槛，使求解器更加通用且易于集成。未来数值计算求解器的发展将着眼于“智能、高效、通用、自主、安全”的技术愿景，推动数值计算从传统的单一应用走向跨学科、多领域的全面应用。具体而言，数值计算求解器将实现以下三大突破：

1) 算力瓶颈突破：通过异构计算、量子计算等新兴技术，解决传统计算架构中的算力瓶颈，实现超大规模计算问题的高效求解。

2) 跨学科技术壁垒突破：通过融合多学科领域的技术，推动求解器在不同应用场景中的广泛适配，提供全方位的解决方案，满足不同行业的个性化需求。

3) 软硬协同技术突破：通过自主研发核心算法与软硬件系统，解决关键领域

的技术瓶颈，提升求解器在全球市场的核心竞争力。

未来，求解器将在高性能计算、人工智能、量子计算等前沿领域取得突破，逐步走向国际市场，成为全球数值计算技术的领先者。随着数值计算求解器技术的不断发展，它将成为推动科学研究、工程创新和智能决策的重要支撑工具，推动全球数字经济和产业技术的进步。

5.3 AI 辅助求解技术发展方向和展望

通用求解器（如天筹、Gurobi、CPLEX、SCIP等）被誉为工业软件“皇冠上的明珠”。当前，这些求解器正经历从传统运筹学算法向“AI+运筹”融合驱动的技术范式演进。基于大模型的自动算法发现技术是当前的研究热点，展现出巨大潜力。在实际工业应用中，该技术仍处于探索阶段，但已为求解器技术的发展指明了新的突破方向。

5.3.1 AI 辅助求解技术发展方向

（一）基于大模型的自动算法设计

这是当前最前沿、最具颠覆性的发展方向。过去，AI在求解器中的应用主要是辅助“预测”（例如预测哪个分支更好），而现在正在转向直接“创造”算法。通过这些基于大模型“创造”得到的算法，实现传统求解器效率瓶颈的突破。

广义上，从小到大，从简单到复杂，基于大模型的算法发现“创造”的对象包括如下四类：

1) 参数级搜索（Parameter Optimization）

- 定义：在既定算法框架下，通过大模型搜索当前场景下的全局最优参数配置。
- 典型应用：超参数自动化调优（Auto-tuning）、启发式算子参数适配。

2) 组件级设计（Component Synthesis）

- 定义：针对算法内部的特定逻辑单元进行选型或生成，实现局部功能的重构与优化。
- 典型应用：算子自动选择、神经网络架构搜索（NAS）、损失函数自主设

计。

3) 算法级生成（Algorithm Generation）

- 定义：面向特定问题域，在不完全依赖现有范式的情况下，从零或从基础逻辑出发生成全新的求解策略。
- 典型应用：自动化数学建模、新型启发式算法设计（Heuristic Design）。

4) 系统级组合（Algorithmic Composition）

- 定义：通过对现有多个组件或独立算法进行高阶逻辑组合与动态选择，构建能够应对复杂任务的集成化求解方案。
- 典型应用：自动化算法选型、多策略融合的启发式框架生成。

通过研究上述四类自动算法发现问题，可以利用大模型，从不同维度优化现有求解器，从而构建出具备“自适应、自演进”能力的下一代AI求解器解决方案。这不仅能打破传统求解器在通用性与高效性之间的零和博弈，还能显著降低求解器技术的应用门槛。

（二）从“静态”向“动态”的进化

传统求解器依赖人工设计的通用规则（如分支策略、剪枝逻辑），这些规则在处理不同分布的示例时往往表现参差不齐。

而利用“参数搜索”（参数级搜索类）和“组件设计”（组件级设计类），大模型可以针对每一个具体的输入实例，动态生成或匹配最优的参数配置与算子组合。这使得求解器不再是一个僵化的黑盒，而是一个能根据“地形”自动切换“装备”的变形金刚，实现对特定工业场景（如仓储物流、电力调度）的极致性能优化。

（三）从“黑盒”向“白盒”的跨越

区别于传统深度强化学习输出难以解释的神经网络权重，基于大模型的“算法生成”（算法级生成类）能够输出可读、可编译、可验证的代码（如Python/C++）。

这意味着AI不仅是在优化求解器，更是在教导人类专家。大模型可能会发现人类未曾设想的启发式逻辑，并将这些隐性知识显性化为代码。这种“白盒优化”让新发现的算法可以被无缝集成到现有的工业级求解器内核中，实现可解释的性能跃升。

（四）从“单一”向“混合”的升维

面对复杂的非凸或大规模组合优化问题，单一算法往往独木难支。

通过“算法组合”（系统级组合类），大模型充当了“超启发式（Hyper-heuristic）大脑”的角色。它可以根据求解过程中的实时反馈（如收敛曲线），在精确算法（MIP）、元启发式（Meta-heuristics）和局部搜索（Local Search）之间进行智能调度与融合。例如，在求解初期使用大模型生成的构造性启发式算法快速热启动，中后期切换至传统求解器进行精确寻优。这种协同机制将改变现有求解器的架构设计。

5.3.2 AI 辅助求解技术展望

站在通用人工智能（Artificial General Intelligence，AGI）与工业 4.0 的交汇点，自动算法设计正经历从“辅助工具”向“自主进化实体”的范式跃迁。展望未来，行业将围绕感知维度、应用深度、进化机制及工程尺度四大核心维度，开启全栈式智能化的新赛道：

1) 多模态融合的深度探索：打破现有算法设计对文本逻辑的单一依赖，探索文本、视觉等多模态信息的特征对齐与跨模态生成，是打造原生多模态算法设计能力的核心。

2) 复杂工业场景的落地应用：致力于解决实际生产中多约束、非线性、高复杂度的算法难题，将算法设计的触角从理想化的学术问题延伸至多元化的工业实际应用。

3) 智能体化（Agentic）框架的范式演进：超越简单的Prompt变换，通过构建具备自我感知与环境反馈能力的Agent框架，实现算法的持续在线进化与零成本迁移。

4) 全流程、工程化的自动演进：突破函数级优化的规模限制，实现针对大型工程级代码体系的自动重构与优化；通过构建自动化的算法评价指标体系，达成真正的端到端、零人工干预设计流程。

AI辅助求解技术在重塑行业格局的同时，也带来了全新的挑战与机遇：

1) 数据资产的竞争壁垒：工业级数学规划实例的规模与质量将成为核心竞争力。收集范围足够广、数量足够多的模型实例，以此来训练专有的求解器大模型（Solver-specific LLMs），将实现从经验沉淀到模型能力的转化。

2) 求解范式的协同变革：形成“AI提议，求解器验证”的创新架构。AI凭借其在大规模数据中习得的“直觉”提供高潜力的候选解，而传统求解器则专注于

可行性校验与最优性证明。这种“启发式预判+确定性验证”的深度耦合，有望成为未来的主流范式。

3) 求解器架构的异构重构：传统单体C++架构正向支持GPU/NPU加速的混合异构架构演进。通过将高频矩阵运算与神经网络推理迁移至算力更强的加速芯片，而将分支定界等复杂逻辑保留在CPU侧，实现计算效率的跨代提升。

综上所述，AI辅助求解器的演进路径可概括为：从“预测参数”向“算法发现”跃迁，从“黑盒加速”向“白盒生成”演进。

特别是基于大模型的算法发现（Algorithm Discovery），极具潜力突破分支定界等经典算法维持数十年的性能瓶颈，将求解效率推向新的数量级。未来，求解器智能体将不仅作为求解工具，更将深度介入建模与优化决策全生命周期，赋能人类探索最优解空间的边界。

参考文献

[1] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks (Vol. 4, pp. 1942-1948). IEEE.

[2] Toro, E. F. (2013). Riemann solvers and numerical methods for fluid dynamics: a practical introduction. Springer Science & Business Media.

[3] Belytschko, T., Liu, W. K., Moran, B., & Elkhodary, K. (2014). Nonlinear finite elements for continua and structures (2nd ed.). John wiley & sons.

[4] Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378, 686-707.

[5] Fletcher, R., & Leyffer, S. (1994). Solving mixed integer nonlinear programs by outer approximation. Mathematical Programming, 66(1), 327-349.

[6] Simpson, T. W., Poplinski, J. D., Koch, P. N., & Allen, J. K. (2001). Metamodels for computer-based engineering design: survey and recommendations. Engineering with Computers, 17(2), 129-150.

[7] Blazek, J. (2015). Computational fluid dynamics: principles and applications (3rd ed.). Butterworth-Heinemann.

[8] Xia, Q., Yang, J., Kim, J., & Li, Y. (2024). On the phase field based model for the crystalline transition and nucleation within the Lagrange multiplier framework. Journal of Computational Physics, 513, 113158.

[9] Lu, S., & Xu, X. (2021). An efficient diffusion generated motion method for wetting dynamics. Journal of Computational Physics, 441, 110476.

[10] Huang, M., Li, R., Yan, K., Yao, C., & Ying, W. (2026). An Efficient Monte Carlo Simulation for Radiation Transport Based on Global Optimal Reference Field. Journal of Computational and Applied Mathematics, 117495.

[11] Bengio, Y., Lodi, A., & Prouvost, A. (2021). Machine learning for combinatorial optimization: A methodological tour d'horizon. European Journal of Operational Research, 290(2), 405-421.

[12] Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2009). ParamILS: An automatic algorithm configuration framework. Journal of Artificial Intelligence Research, 36, 267-306.

[13] Huang, Y., Xu, L., & Hutter, F. (2024). Learning to configure solvers: A comprehensive survey on automated algorithm configuration. Journal of Artificial Intelligence Research, 79, 1125-1189.

[14] Pinedo, M. L. (2022). Scheduling: Theory, Algorithms, and Systems (6th ed.). Springer.

[15] Achterberg, T., & Wunderling, R. (2013). Mixed integer programming: Analyzing 12 years of progress. In Facets of combinatorial optimization: Festschrift for martin grötschel (pp. 449-481). Berlin, Heidelberg: Springer Berlin Heidelberg.

[16] Gurobi Optimization, LLC. (2022). The Total Economic Impact™ of Gurobi Optimizer. Forrester Research.

[17] Kotary, J., Fioretto, F., Van Hentenryck, P., & Dilkina, B. (2021). End-to-end constrained optimization learning: A survey. IEEE Transactions on Neural Networks and Learning Systems,

34(10), 6960-6980.

[18] Balay, S., Gropp, W. D., McInnes, L. C., & Smith, B. F. (1997). Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, & H. P. Langtangen (Eds.), Modern Software Tools in Scientific Computing (pp. 163–202). Birkhäuser Press.

[19] Falgout, R. D., & Yang, U. M. (2002). hypre: A library of high performance preconditioners. In International Conference on computational science (pp. 632-641). Berlin, Heidelberg: Springer Berlin Heidelberg.

[20] Blanchard, P., Higham, N. J., & Tisseur, F. (2021). The accuracy of linear solvers in finite precision arithmetic. SIAM Journal on Matrix Analysis and Applications, 42(4), 1683-1705.

[21] Bach, F. (2023). High-dimensional optimization: The curse of dimensionality and beyond. Foundations and Trends® in Machine Learning, 16(2), 123-245.

[22] Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., & Jordan, M. I. (2021). How to escape saddle points efficiently. Communications of the ACM, 64(10), 96-104.

[23] Demmel, J. (2020). Communication-avoiding algorithms for linear algebra and beyond. Proceedings of the IEEE, 108(11), 1928-1943.

[24] Intriligator, M. D. (2002). Mathematical optimization and economic theory. Society for Industrial and Applied Mathematics.

[25] Gershenfeld, N. A. (1999). The nature of mathematical modeling. Cambridge university press.

[26] Nocedal, J., & Wright, S. J. (2006). Numerical optimization. New York, NY: Springer New York.

[27] Bertsimas, D., & Tsitsiklis, J. N. (1997). Introduction to linear optimization. Athena Scientific.

[28] Dantzig, G. B. (2016). Linear programming and extensions.

[29] Wright, S. J. (1997). Primal-dual interior-point methods. Society for Industrial and Applied Mathematics (SIAM).

[30] Goldfarb, D., & Idnani, A. (1983). A numerically stable dual method for solving strictly convex quadratic programs. Mathematical Programming, 27(1), 1–33.

[31] Alizadeh, F., & Goldfarb, D. (2003). Second-order cone programming. Mathematical Programming, 95(1), 3–51.

[32] Bertsekas, D. P. (2016). Nonlinear programming (3rd ed.). Athena Scientific. MIT Source.

[33] Bazaraa, M. S., Sherali, H. D., & Shetty, C. M. (2006). Nonlinear programming: Theory and algorithms (3rd ed.). John Wiley & Sons. Wiley Online Library.

[34] Luenberger, D. G., & Ye, Y. (2015). Linear and nonlinear programming (4th ed.). Springer. Stanford University.

[35] Wolsey, L. A. (1998). Integer programming. Wiley.

[36] Land, A. H., & Doig, A. G. (1960). An automatic method of solving discrete programming problems. Econometrica, 28(3), 497–520.

[37] Achterberg, T., Bixby, R. E., Gu, Z., Rothberg, E., & Wesselmann, E. (2020). Presolve reductions in mixed integer programming. INFORMS Journal on Computing, 32(2), 473–506.

[38] Savelsbergh, M. W. P. (1994). Preprocessing and probing techniques for mixed integer programming problems. ORSA Journal on Computing, 6(4), 445–454.

[39] Linderoth, J. T., & Savelsbergh, M. W. P. (1999). A computational study of search strategies for mixed integer programming. INFORMS Journal on Computing, 11(2), 173–187.

[40] Achterberg, T. (2009). Constraint integer programming (Doctoral dissertation, Zuse Institute

Berlin (ZIB).

[41] Berthold, T. (2006). Primal heuristics for mixed integer programs (Doctoral dissertation, Zuse Institute Berlin (ZIB).

[42] Danna, E., Rothberg, E., & Le Pape, C. (2005). Exploring relaxation induced neighborhoods to improve MIP solutions. Mathematical Programming, 102(1), 71–90.

[43] Gomory, R. E. (1958). Outline of an algorithm for integer solutions to linear programs. Bulletin of the American Mathematical Society, 64(5), 275–278.

[44] Marchand, H., Martin, A., Weismantel, R., & Wolsey, L. A. (2002). Cutting planes in integer and mixed integer programming. Discrete Applied Mathematics, 123(1-3), 397–446.

[45] Bienstock, D. (1996). Computational study of a family of mixed-integer quadratic programming problems. Mathematical Programming, 74(2), 121–140.

[46] Stubbs, R. A., & Mehrotra, S. (1999). A branch-and-cut algorithm for 0-1 mixed convex programming. Mathematical Programming, 86(3), 515–532.

[47] Rossi, F., Van Beek, P., & Walsh, T. (Eds.). (2006). Handbook of constraint programming. Elsevier.

[48] Apt, K. (2003). Principles of constraint programming. Cambridge University Press.

[49] Stuckey, P. J. (2010). Lazy clause generation: Combining the power of SAT and CP (and MIP?) solving. In International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming (pp. 5-9). Berlin, Heidelberg: Springer Berlin Heidelberg.

[50] Ohrimenko, O., Stuckey, P. J., & Codish, M. (2009). Propagation via lazy clause generation. Constraints, 14(3), 357-391.

[51] Biere, A., van Maaren, H., & Walsh, T. (Eds.). (2009). Handbook of satisfiability. SAGE Publications Limited.

[52] Silva, J. M., & Sakallah, K. A. (1996). GRASP-a new search algorithm for satisfiability. In Proceedings of International Conference on Computer Aided Design (pp. 220-227). IEEE.

[53] Moskewicz, M. W., Madigan, C. F., Zhao, Y., Zhang, L., & Malik, S. (2001). Chaff: Engineering an efficient SAT solver. In Proceedings of the 38th annual Design Automation Conference (pp. 530-535).

[54] Liang, J., Ganesh, V., Poupart, P., & Czarnecki, K. (2016). Exponential recency weighted average branching heuristic for SAT solvers. In Proceedings of the AAAI Conference on Artificial Intelligence 30(1).

[55] Biere, A., & Fröhlich, A. (2015). Evaluating CDCL variable scoring schemes. In International conference on theory and applications of satisfiability testing (pp. 405-422). Cham: Springer International Publishing.

[56] McKay, M. D., Beckman, R. J., & Conover, W. J. (2000). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics, 42(1), 55-61.

[57] Ackley, D. (2012). A connectionist machine for genetic hillclimbing. Springer science & business media.

[58] Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary computation, 8(2), 173-195.

[59] Williams, C. K., & Rasmussen, C. E. (2006). Gaussian processes for machine learning (Vol. 2,

No. 3, p. 4). Cambridge, MA: MIT press.

[60] Paria, B., Kandasamy, K., & Póczos, B. (2020, August). A flexible framework for multi-objective bayesian optimization using random scalarizations. In Uncertainty in Artificial Intelligence (pp. 766-776). PMLR.

[61] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

[62] Sastry, K., Goldberg, D. E., & Kendall, G. (2013). Genetic algorithms. In Search methodologies: Introductory tutorials in optimization and decision support techniques (pp. 93-117). Boston, MA: Springer US.

[63] Deb, K., & Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. Complex systems, 9(2), 115-148.

[64] Deb, K., & Goyal, M. (1996). A combined genetic adaptive search (GeneAS) for engineering design. Computer Science and informatics, 26, 30-45.

[65] Torczon, V. (1997). On the convergence of pattern search algorithms. SIAM Journal on optimization, 7(1), 1-25.

[66] Golub, G. H., & Van Loan, C. F. (2013). Matrix Computations (4th ed.). Johns Hopkins University Press.

[67] Toselli, A., & Widlund, O. (2005). Domain decomposition methods: Algorithms and theory. Springer Science & Business Media.

[68] Ruge, J. W., & Stüben, K. (1987). Multigrid methods. Frontiers in Applied Mathematics, 3, 73-130.

[69] Demmel, J. W., Eisenstat, S. C., Gilbert, J. R., Li, X. S., & Liu, J. W. H. (1999). A supernodal approach to sparse partial pivoting. SIAM Journal on Matrix Analysis and Applications, 20(3), 720-755.

[70] Liu, J. W. H. (1990). The role of elimination trees in sparse factorization. SIAM Journal on Matrix Analysis and Applications, 11(1), 134-172.

[71] Anzt, H., Lichev, S., & Flegar, G. (2020). Mixed precision algebraic multigrid on GPUs. Euro-Par 2020: Parallel Processing, 452-466.

[72] Rose, D. J. (1970). Triangulated graphs and the elimination process. Journal of Mathematical Analysis and Applications, 32(3), 597-609.

[73] George, A., & Liu, J. W. H. (1989). The evolution of the minimum degree ordering algorithm. SIAM Review, 31(1), 1-19.

[74] George, A. (1973). Nested dissection of a regular finite element mesh. SIAM Journal on Numerical Analysis, 10(2), 345-363.

[75] Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., & Van der Vorst, H. (1994). Templates for the solution of linear systems: Building blocks for iterative methods. SIAM.

[76] Ames W F. (2014). Numerical methods for partial differential equations. Academic press.

[77] Moré, J. J., & Thuente, D. J. (1994). Line search algorithms with guaranteed sufficient decrease. ACM Transactions on Mathematical Software (TOMS), 20(3), 286-307.

[78] Sorensen, D. C. (1992). Implicit application of polynomial filters in ak-step Arnoldi method. Siam journal on matrix analysis and applications, 13(1), 357-385.

[79] Campos, C., & Roman, J. E. (2012). Strategies for spectrum slicing based on restarted Lanczos

methods. Numerical Algorithms, 60(2), 279-295.

[80] Polizzi, E. (2009). Density-matrix-based algorithm for solving eigenvalue problems. Physical Review B—Condensed Matter and Materials Physics, 79(11), 115112.

[81] Bennighof, J. K., & Lehoucq, R. B. (2004). An automated multilevel substructuring method for eigenspace computation in linear elastodynamics. SIAM Journal on Scientific Computing, 25(6), 2084-2106.

[82] Duersch, J. A., Shao, M., Yang, C., & Gu, M. (2018). A robust and efficient implementation of LOBPCG. SIAM Journal on Scientific Computing, 40(5), C655-C676.

[83] Li, X., Zhu, F., Zhen, H. L., Luo, W., Lu, M., Huang, Y., ... & Mao, K. (2024). Machine learning insides optverse ai solver: Design principles and applications. arXiv preprint arXiv:2401.05960.

[84] Cowen-Rivers, A. I., Lyu, W., Tutunov, R., Wang, Z., Grosnit, A., Griffiths, R. R., ... & Bou-Ammar, H. (2022). Hebo: Pushing the limits of sample-efficient hyper-parameter optimisation. Journal of Artificial Intelligence Research, 74, 1269-1349.

[85] Lawless, C., Li, Y., Wikum, A., Udell, M., & Vitercik, E. (2024). LLMs for cold-start cutting plane separator configuration. arXiv preprint arXiv:2412.12038.

[86] Yilmaz, K., & Yorke-Smith, N. (2021). A study of learning search approximation in mixed integer branch and bound: Node selection in scip. Ai, 2(2), 150-178.

[87] Gupta, P., Gasse, M., Khalil, E., Mudigonda, P., Lodi, A., & Bengio, Y. (2020). Hybrid models for learning to branch. Advances in neural information processing systems, 33, 18087-18097.

[88] Kuang, Y., Wang, J., Liu, H., Zhu, F., Li, X., Zeng, J., ... & Wu, F. (2024). Rethinking branching on exact combinatorial optimization solver: The first deep symbolic discovery framework. In The Twelfth International Conference on Learning Representations.

[89] Kuang, Y., Wang, J., Zhou, Y., Li, X., Zhu, F., Hao, J., & Wu, F. (2024, July). Towards general algorithm discovery for combinatorial optimization: Learning symbolic branching policy from bipartite graph. In Forty-first International Conference on Machine Learning.

[90] Wang, J., Wang, Z., Li, X., Kuang, Y., Shi, Z., Zhu, F., ... & Wu, F. (2024). Learning to cut via hierarchical sequence/set model for efficient mixed-integer programming. IEEE Transactions on Pattern Analysis and Machine Intelligence.

[91] Ye, M., Wang, J., Zhu, F., Wang, Z., Kuang, Y., Li, X., ... & Wu, F. (2025). Dynamic Configuration for Cutting Plane Separators via Reinforcement Learning on Incremental Graph. In The Thirty-ninth Annual Conference on Neural Information Processing Systems.

[92] Nair, V., Bartunov, S., Gimeno, F., Von Glehn, I., Lichocki, P., Lobov, I., ... & Zwols, Y. (2020). Solving mixed integer programs using neural networks. arXiv preprint arXiv:2012.13349.

[93] Han, Q., Yang, L., Chen, Q., Zhou, X., Zhang, D., Wang, A., ... & Luo, X. (2023). A gnn-guided predict-and-search framework for mixed-integer linear programming. arXiv preprint arXiv:2302.05636.

[94] Liu, H., Wang, J., Geng, Z., Li, X., Zong, Y., Zhu, F., ... & Wu, F. (2025). Apollo-MILP: An alternating prediction-correction neural solving framework for mixed-integer linear programming. arXiv preprint arXiv:2503.01129.

[95] Li, S., Yu, Y., Deng, Y., Zhang, Z., Chen, M., Zhu, F., ... & An, B. (2025). Collab-Solver: Collaborative Solving Policy Learning for Mixed-Integer Linear Programming. arXiv preprint arXiv:2508.03030.

[96] Zhou, Y., Wang, J., Kuang, Y., Li, X., Luo, W., HAO, J., & Wu, F. (2024). LLM4Solver: Large

Language Model for Efficient Algorithm Design of Combinatorial Optimization Solver.

[97] Yazdani, M., Mostajabdaveh, M., Aref, S., & Zhou, Z. (2025). EvoCut: Strengthening Integer Programs via Evolution-Guided Language Models. arXiv preprint arXiv:2508.11850

[98] Buschkühl, L., Sahling, F., Helber, S., & Tempelmeier, H. (2010). Dynamic capacitated lot-sizing problems: a classification and review of solution approaches. Or Spectrum, 32(2), 231-261.

[99] Copil, K., Wörbelauer, M., Meyr, H., & Tempelmeier, H. (2017). Simultaneous lotsizing and scheduling problems: a classification and review of models. OR spectrum, 39(1), 1-64.

[100] Sambasivan, M., & Yahya, S. (2005). A Lagrangean-based heuristic for multi-plant, multi-item, multi-period capacitated lot-sizing problems with inter-plant transfers. Computers & Operations Research, 32(3), 537-555.

[101] Cunha, J. O., Kramer, H. H., & Melo, R. A. (2019). Effective matheuristics for the multi-item capacitated lot-sizing problem with remanufacturing. Computers & Operations Research, 104, 149-158.

[102] Wu, T., Shi, L., Geunes, J., & Akartunalı, K. (2011). An optimization framework for solving capacitated multi-level lot-sizing problems with backlogging. European Journal of Operational Research, 214(2), 428-441.

[103] Seeanner, F., Almada-Lobo, B., & Meyr, H. (2013). Combining the principles of variable neighborhood decomposition search and the fix&optimize heuristic to solve multi-level lot-sizing and scheduling problems. Computers & Operations Research, 40(1), 303-317.

[104] Toledo, C. F. M., De Oliveira, R. R. R., & França, P. M. (2013). A hybrid multi-population genetic algorithm applied to solve the multi-level capacitated lot sizing problem with backlogging. Computers & Operations Research, 40(4), 910-919.

[105] Toledo, C. F. M., da Silva Arantes, M., Hossomi, M. Y. B., França, P. M., & Akartunalı, K. (2015). A relax-and-fix with fix-and-optimize heuristic applied to multi-level lot-sizing problems. Journal of heuristics, 21(5), 687-717.

[106] Xiao, J., Zhang, C., Zheng, L., & Gupta, J. N. (2013). MIP-based fix-and-optimise algorithms for the parallel machine capacitated lot-sizing and scheduling problem. International Journal of Production Research, 51(16), 5011-5028.

[107] Chen, H. (2015). Fix-and-optimize and variable neighborhood search approaches for multi-level capacitated lot sizing problems. Omega, 56, 25-36.

[108] Nascimento, M. C., Resende, M. G., & Toledo, F. M. (2010). GRASP heuristic with path-relinking for the multi-plant capacitated lot sizing problem. European Journal of Operational Research, 200(3), 747-754.

[109] Carvalho, D. M., & Nascimento, M. C. (2022). Hybrid matheuristics to solve the integrated lot sizing and scheduling problem on parallel machines with sequence-dependent and non-triangular setup. European Journal of Operational Research, 296(1), 158-173.

[110] Dziuba, D., & Almeder, C. (2023). New construction heuristic for capacitated lot sizing problems. European Journal of Operational Research, 311(3), 906-920.

[111] Balas, E. (1969). Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. Operations research, 17(6), 941-957.

[112] Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. Management science, 34(3), 391-401.

[113] Taillard, É. D. (1994). Parallel taboo search techniques for the job shop scheduling problem. ORSA journal on Computing, 6(2), 108-117.

[114] Li, R., Gong, W., Lu, C., & Wang, L. (2022). A learning-based memetic algorithm for energy-efficient flexible job-shop scheduling with type-2 fuzzy processing time. IEEE Transactions on Evolutionary Computation, 27(3), 610-620.

[115] Zeiträg, Y., Rui Figueira, J., & Figueira, G. (2024). A cooperative coevolutionary hyper-heuristic approach to solve lot-sizing and job shop scheduling problems using genetic programming. International Journal of Production Research, 62(16), 5850-5877.

[116] Toth, P., & Vigo, D. (Eds.). (2002). The vehicle routing problem. Society for Industrial and Applied Mathematics.

[117] Erdoğan, S., & Miller-Hooks, E. (2012). A green vehicle routing problem. Transportation research part E: logistics and transportation review, 48(1), 100-114.

[118] Crevier, B., Cordeau, J. F., & Laporte, G. (2007). The multi-depot vehicle routing problem with inter-depot routes. European journal of operational research, 176(2), 756-773.

[119] Longo, H., De Aragao, M. P., & Uchoa, E. (2006). Solving capacitated arc routing problems using a transformation to the CVRP. Computers & Operations Research, 33(6), 1823-1837.

[120] Desaulniers, G., Desrosiers, J., Erdmann, A., Solomon, M. M., & Soumis, F. (2002). VRP with Pickup and Delivery. The vehicle routing problem, 9, 225-242.

[121] Braekers, K., Ramaekers, K., & Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. Computers & industrial engineering, 99, 300-313.

[122] Eksioglu, B., Vural, A. V., & Reisman, A. (2009). The vehicle routing problem: A taxonomic review. Computers & Industrial Engineering, 57(4), 1472-1483.

[123] Caceres-Cruz, J., Arias, P., Guimarans, D., Riera, D., & Juan, A. A. (2014). Rich vehicle routing problem: Survey. ACM Computing Surveys (CSUR), 47(2), 1-28.

[124] Konstantakopoulos, G. D., Gayialis, S. P., & Kechagias, E. P. (2022). Vehicle routing problem and related algorithms for logistics distribution: a literature review and classification: GD Konstantakopoulos et al. Operational research, 22(3), 2033-2062.

[125] Wäscher, G., Haußner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. European journal of operational research, 183(3), 1109-1130.

[126] Heckmann, R., & Lengauer, T. (1995). A simulated annealing approach to the nesting problem in the textile manufacturing industry. Annals of Operations Research, 57(1), 103-133.

[127] Umetani, S., Yagiura, M., Imahori, S., Imamichi, T., Nonobe, K., & Ibaraki, T. (2009). Solving the irregular strip packing problem via guided local search for overlap minimization. International Transactions in Operational Research, 16(6), 661-683.

[128] Gomes, A. M., & Oliveira, J. F. (2002). A 2-exchange heuristic for nesting problems. European Journal of Operational Research, 141(2), 359-370.

[129] Elkeran, A. (2013). A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering. European Journal of Operational Research, 231(3), 757-769.

[130] 刘胡瑶. (2007). 基于临界多边形的二维排样算法研究(博士学位论文, 上海交通大学). 博士

[131] Cherri, L. H., Mundim, L. R., Andretta, M., Toledo, F. M., Oliveira, J. F., & Carravilla, M. A. (2016). Robust mixed-integer linear programming models for the irregular strip packing problem. European Journal of Operational Research, 253(3), 570-583.

[132] Oliveira, J. F., Gomes, A. M., & Ferreira, J. S. (2000). TOPOS–A new constructive algorithm for nesting problems: TOPOS–Ein neues Konstruktionsverfahren für das „Nesting-Problem". OR-Spektrum, 22(2), 263-284.

[133] Dyckhoff, H. (1990). A typology of cutting and packing problems. European journal of operational research, 44(2), 145-159.

[134] Bortfeldt, A., & Wäscher, G. (2013). Constraints in container loading–a state-of-the-art review. European Journal of Operational Research, 229(1), 1-20.

[135] Gajda, M., Trivella, A., Mansini, R., & Pisinger, D. (2022). An optimization approach for a complex real-life container loading problem. Omega, 107, 102559.

[136] Eley, M. (2002). Solving container loading problems by block arrangement. European Journal of Operational Research, 141(2), 393-409.

[137] 张德富, 彭煜 & 张丽丽. (2012). 求解三维装箱问题的多层启发式搜索算法. 计算机学报, 35(12), 2553-2561.

[138] Crainic, T. G., Perboli, G., & Tadei, R. (2008). Extreme point-based heuristics for three-dimensional bin packing. Informs Journal on computing, 20(3), 368-384.

[139] Heßler, K., Hintsch, T., & Wienkamp, L. (2025). A fast optimization approach for a complex real-life 3D Multiple Bin Size Bin Packing Problem. European Journal of Operational Research.

[140] Abdelghany, A., & Abdelghany, K. (2019). Airline Network Planning and Scheduling. John Wiley & Sons.

[141] Zhang, S., Wu, L., Yan, W., & Ji, Z. (2020). Vulcanization Shop Scheduling Based on Distribution Estimation Algorithm. Journal of System Simulation, 29(9), 2182-2189.

[142] Yu, S., Yang, D., Wang, X., Zhu, K., & Zheng, B. (2011, May). A two-stage heuristic method for vulcanization production scheduling. In 2011 Chinese Control and Decision Conference (CCDC) (pp. 3601-3605). IEEE.

[143] Chen, H. W., Li, K., Tan, Y. Q., Jia, F., & Chen, L. (2025). Traditional Chinese medicines: a supply chain management perspective. International Journal of Logistics Research and Applications, 1-20.

[144] Chen, C. N., Lai, C. H., Lu, G. W., Huang, C. C., Wu, L. J., Lin, H. C., & Chen, P. S. (2022, March). Applying simulation optimization to minimize drug inventory costs: a study of a case outpatient pharmacy. In Healthcare (Vol. 10, No. 3, p. 556). MDPI.

[145] 尹晓辉. (2010). 铝合金冷轧及薄板生产技术. 冶金工业出版社.

[146] Dauzère-Pérès, S., Ding, J., Shen, L., & Tamssaouet, K. (2024). The flexible job shop scheduling problem: A review. European Journal of Operational Research, 314(2), 409-432.

[147] D. Kopp, M. Hassoun, & A. Kalir and L. Mönch. (2020). Integrating Critical Queue Time Constraints Into SMT2020 Simulation Models. Winter Simulation Conference.

[148] Liu, S. Q., & Kozan, E. (2009). Scheduling a flow shop with combined buffer conditions. International Journal of Production Economics.

[149] Kopanos, G. M., Puigjaner, L., & Maravelias, C. T. (2011). Production planning and scheduling of parallel continuous processes with product families. Industrial & engineering chemistry research, 50(3), 1369-1378.

[150] Menezes, A. A., Clark, A., & Almada-Lobo, B. (2011). Capacitated lot-sizing and scheduling with sequence-dependent, period-overlapping and non-triangular setups. Journal of Scheduling, 14(2), 209-219.

[151] Xiao, J., Zhang, C., Zheng, L., & Gupta, J. N. (2013). MIP-based fix-and-optimise algorithms for the parallel machine capacitated lot-sizing and scheduling problem. International Journal of Production Research, 51(16), 5011-5028.

[152] Guimarães, L., Klabjan, D., & Almada-Lobo, B. (2014). Modeling lotsizing and scheduling problems with sequence dependent setups. European Journal of Operational Research, 239(3), 644-662.

[153] 郑少峰. (2026). 物流配送, 路径优化技术研究综述及前沿概述——基于 CiteSpace 的可视化分析. World Economic Research, 15, 164.

[154] Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., & Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research, 60(3), 611-624.

[155] Vidal, T. (2022). Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. Computers & Operations Research, 140, 105643.

[156] Santini, A., Ropke, S., & Hvattum, L. M. (2018). A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. Journal of Heuristics, 24(5), 783-815.

[157] Accorsi, L., & Vigo, D. (2021). A fast and scalable heuristic for the solution of large-scale capacitated vehicle routing problems. Transportation Science, 55(4), 832-856.

[158] Penna, P. H. V., Subramanian, A., & Ochi, L. S. (2013). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. Journal of Heuristics, 19(2), 201-232.

[159] Anderson, J. D., & Wendt, J. (1995). Computational fluid dynamics (Vol. 206, p. 332). New York: McGraw-hill.

[160] 张来平, 常兴华 & 赵钟等. (2017). 计算流体力学网格生成技术. 科学出版社.

[161] Launder, B. E., & Spalding, D. B. (1983). The numerical computation of turbulent flows. In Numerical prediction of flow, heat transfer, turbulence and combustion (pp. 96-116). Pergamon.

[162] Li, S., Kulkarni, J., Menache, I., Wu, C., & Li, B. (2024). Towards foundation models for mixed integer linear programming. arXiv preprint arXiv:2410.08288.

[163] Huang, P., Wu, Y., Ma, Y., Wu, C., Song, W., & Zhang, W. (2026). A General Neural Backbone for Mixed-Integer Linear Optimization via Dual Attention. arXiv preprint arXiv:2601.04509.

附录A：缩略语表

英文缩写	英文全称	中文全称
ADMM	Alternative Direction Method of Multipliers	交替方向乘子法
AMG	Algebraic Multigrid	代数多重网格法
AMLS	Automated Multi-Level Substructuring	自动多级子结构法
BFGS	Broyden-Fletcher-Goldfarb-Shanno	BFGS 算法（拟牛顿法的一种）
BiCG	Biconjugate Gradient	双共轭梯度法
BiCGStab	Biconjugate Gradient Stabilized	稳定化双共轭梯度法
BLAS	Basic Linear Algebra Subprograms	基础线性代数子程序标准库
BO	Black-Box Optimization	黑箱优化
BVP	Boundary Value Problem	边值问题
CAMG	Classical Algebraic Multigrid	经典代数多重网格法
CDCL	Conflict Driven Clause Learning	冲突分析子句学习
CFL	Courant-Friedrichs-Lewy Condition	CFL 条件（稳定性判据）
CG	Conjugate Gradient	共轭梯度法
CIP	Constraint Integer Programming	约束整数规划
COO	Coordinate	坐标格式
CP	Constraint Programming	约束规划
CPU	Central Processing Unit	中央处理器
CSC	Compressed Sparse Column	压缩稀疏列格式
CSR	Compressed Sparse Row	压缩稀疏行格式
FDM	Finite Difference Method	有限差分法
FEAST	Fast Eigenvalue Algorithm for Spectral Transformations	基于围道积分的高性能特征值求解算法
FEM	Finite Element Method	有限元法
FVM	Finite Volume Method	有限体积法
GE	Gaussian Elimination	高斯消元法
GMRES	Generalized Minimum RESidual	广义最小残差法
GNN	Graph Neural Network	图神经网络
GPBiCG	Generalized Product Biconjugate Gradient	广义乘积双共轭梯度法
GPT	Generative Pre-trained Transformer	生成式预训练模型

GPU	Graphics Processing Unit	图形处理器
GTE	Global Truncation Error	整体截断误差
ICC	Incomplete Cholesky Composition	不完全Cholesky分解
ICCK	Incomplete Cholesky Composition with K-level fill	K阶不完全Cholesky
IIS	Irreducible Infeasible Subsystem	不可约不可行子系统
ILP	Integer Linear Programming	整数线性规划
ILU	Incomplete Lower-Upper	不完全下-上三角分解
ILUK	Incomplete LU factorization with K-level fill	K阶不完全LU分解
ILUTP	Incomplete LU with Threshold and Pivoting	带阈值和列选主元的ILU分解
IVP	Initial Value Problem	初值问题
KKT	Karush-Kuhn-Tucker Conditions	KKT 条件（最优性条件）
KSP	Krylov Subspace	克雷洛夫子空间
LDLt	Lower-Diagonal-Lower Transpose Decomposition	下三角-对角-下三角转置矩阵分解
LLM	Large Language Model	大语言模型
LOBPCG	Locally Optimal Block Preconditioned Conjugate Gradient	局部最优块预条件共轭梯度法
LP	Linear Programming	线性规划
LS	Least Squares	最小二乘法
LTE	Local Truncation Error	局部截断误差
LU	Lower-Upper Decomposition	LU 分解（下上三角分解）
MaxSAT	Maximum Satisfiablity Problem	最大可满足性问题
MILP	Mixed Integer Linear Programming	混合整数线性规划
MINRES	Minimum Residual	最小残差法
MIQP	Mixed Integer Quadratic Programming	混合整数二次规划
MLLM	Multimodal Large Language Model	多模态大模型
MP	Mathematical Programming	数学规划
MPI	Message Passing Interface	消息传递接口
NLP	Nonlinear Programming	非线性规划
NP-hard	Non-deterministic Polynomial-time hard	非确定性多项式时间难度
NPU	Neural Processing Unit	神经网络处理器
ODE	Ordinary Differential Equation	常微分方程
PBO	Pseudo Boolean Optimization	伪布尔优化
PC	Preconditioner / Preconditioning	预条件算子 / 预条件技术
PCA	Principal Component Analysis	主成分分析

PDE	Partial Differential Equation	偏微分方程
QCQP	Quadratic Constrained Quadratic Programming	二次约束二次规划
RK	Runge-Kutta Methods	龙格-库塔法
SAT	Satisfiablity Problem	可满足性问题
SDK	Software Development Kit	软件开发套件
SIMD	Single Instruction Multiple Data	单指令多数据
SOCP	Second Order Cone Programming	二阶锥规划
SOR	Successive Over-Relaxation	逐次超松弛迭代法
SOTA	State Of The Art	最先进水平
SPD	Symmetric Positive Definite	对称正定（矩阵）
SQP	Sequential Quadratic Programming	序列二次规划
SVD	Singular Value Decomposition	奇异值分解

附录B：术语定义表

B.1 数学规划术语

术语名称	术语解释
优化模型（Optimization Model）	包含决策变量、约束条件和目标函数的数学框架。决策变量是待确定取值的元素。约束条件指定了决策变量的取值范围。目标函数指定了决策变量的取值偏好，在满足约束条件的前提下需要使得目标函数的取值尽可能大或小。
优化方向（Optimization Direction）	优化问题目标函数的取值倾向。其分为极大化（Maximize）方向，即希望目标函数取值尽可能大，和极小化（Minimize）方向，即希望目标函数取值尽可能小。
求解器SDK（Solver SDK）	包含求解器公开接口和运行库的软件包，可用于实现数学优化问题建模、求解、结果获取、不可行分析、回调等功能。
MPS和LP文件（MPS and LP File）	求解器通用的存储数学规划问题的模型和数据文件。支持表示线性规划、混合整数规划、二次规划等数学规划问题。MPS文件以列的格式存储约束矩阵，LP文件以行的格式存储约束矩阵。
SOL文件（SOL File）	记录可行解或最优解的文件。通常记录目标函数取值、决策变量名和决策变量的取值。即使文件扩展名均为sol，不同求解器可能有不同的文件格式定义。
凸问题和非凸问题（Convex and Nonconvex Problem）	数学规划问题的决策变量可行域为凸集或非凸集。凸问题的局部最优解必定为全局最优解。
求解状态（Solving Status）	优化问题已被证明的可行解存在状态。通常分为“最优”（Optimal），即至少一个最优解已找到且已证明不存在目标值更优的解；“不可行”（Infeasible），即已证明不存在任何可行解；“无界”（Unbounded），即存在可行解使得目标函数的取值任意大（对于极大化问题）或任意小（对于极小化问题）；“不可行或无界”（Infeasible or Unbounded），即已证明仅可能是不可行或无界状态。通常称存在可行解的优化问题为可行问题（Feasible Problem）。对应的，不存在可行解的优化问题称为不可行问题（Infeasible Problem）。
确定性并行算法（Deterministic Parallel Algorithm）	能够保持在相同运行环境、相同输入数据和参数下，多次运行结果确定的并行算法。在可能存在多个全局最优解的线性规划、混合整数线性规划等问题中，确定性并行算法能够保证每次运行输出结果均为相同的解。与之相对的，非确定性并行算法（Non-deterministic Parallel Algorithm）的计算结果则存在较大的随机性，但计算效率通常更高。
分支定界与相对最	通过“分而治之”的思想，不断地将数学规划问题决策变量（通常是整

优间隙（Branch-and-Bound and Relative Optimality Gap）	数型变量）的可行域分成两份或多份。分支定界过程中，通过求解松弛问题可以得到原始问题（假设为极小化问题）的最优目标值下界，也称为对偶界（Dual Bound）。原始问题的可行解可以作为原始问题最优目标值的上界，也称为原始界（Primal Bound）。相对最优间隙衡量了当前已获取的最优可行解，也称为候选解（Incumbent），与理论全局最优解的相对距离，也反映了分支定界算法的执行进度，具体计算公式为 $$\text{相对最优间隙} = \frac{\left\lvert \text{可行解} - \text{对偶界} \right\rvert}{\left\lvert \text{可行解} \right\rvert}$$
不可行分析（Infeasibility Analysis）	当优化问题被证明为不可行状态时，用于分析导致不可行的根因约束，或改造约束条件得到可行问题。前者对应于搜索不可约不可行子系统（Irreducible Infeasible Subsystem，IIS），后者对应于可行性松弛（Feasible Relaxation）。IIS是原问题约束条件的子集组成的不可行问题，且再删除任何一条约束条件即变为可行问题。
求解器参数（Solver Parameter）	用于控制求解器算法执行策略、资源使用、最优性和可行性容忍度、终止条件等。
回调（Callback）	使求解器用户能够植入自定义算法、控制逻辑等自定义代码运行逻辑的接口。通常用于植入用户自定义的启发式、割平面算法，用于提升求解器处理某类优化问题的能力。
可行解池（Feasible Solution Pool）	保存求解过程中找到的若干可行解的存储空间。
热启动（Warm Start）	利用某些初始输入信息加速算法执行的过程。例如线性规划单纯形法在求解一系列相近优化问题时，利用前一问题的最优基信息可显著加速单纯形算法并减少迭代次数；混合整数规划的分支定界方法也可利用初始输入的可行解增强剪枝过程，减少不必要的节点搜索。

B.2 约束规划术语

术语名称	术语解释
子句（Clause）	子句是可满足性问题SAT中的常用术语，特指布尔或约束，例如子句 $a \vee b \vee c$ 限制了布尔变量 a, b, c 不能同时为0。
互异约束（All Different Constraint）	互异约束要求一组变量的取值必须各不相同，是约束规划领域中非常典型且实用的一种全局约束。
表约束（Table Constraint）	表约束通过显式枚举允许的变量值组合（如 (x, y, z) 的合法三元组列表）定义关系。适用于复杂或非算术关系（如产品配置规则）。求解器使用紧凑数据结构（如多值决策图）高效检查合法性。
元素约束（Element	元素约束通过索引访问数组（如 $y = \text{array}[x]$），其中 x 是变量，且 $x \in \{1, 2\}$，若 $\text{array} = [10, 20, 30]$，则有 $y \in \{20,\ 30\}$。需特殊推理

Constraint）	维护 x 和 y 的域一致性。
互逆约束（Inverse Constraint）	互逆约束定义两组变量间的双向映射（如 $x_i = j \Leftrightarrow y_j = i$）。常见于调度或排列问题（如任务-机器分配）。求解器需同步推理两组变量的域，确保无冲突。
区间约束（Interval Constraint）	区间约束用于定义区间关系（如 $start + size = end$）。用于时间安排或资源占用问题，推理时需处理边界交叠情况。
无重叠约束（No Overlap Constraint）	无重叠约束要求多个区间（如时间、空间）互不重叠。例如，任务调度中两个任务不能同时占用同一资源。通过端点排序或累加资源约束实现，是调度问题的核心约束。
累积约束（Cumulative Constraint）	累积约束限制资源使用总量（如 $\sum_i r_i \leq R$）。常用于装箱、负载均衡等问题。
资源池约束（Reservoir Constraint）	资源池约束管理共享资源的动态分配（如多个任务共享有限机器池）。需确保资源请求总和不超过池容量，并通过冲突检测避免死锁。常见于柔性制造系统建模。
环路约束（Circuit Constraint）	环路约束要求变量排列构成单个环路（如旅行商问题中的哈密顿回路）。通过禁止子环路和邻接推理实现，是路径优化的关键约束。
路径约束（Routes Constraint）	专用于车辆路径规划（VRP）问题的约束类型，其中包括路径的起点、终点、途经点、容量、需求等等成员。需结合图算法进行推理。
FZN格式（FlatZinc Format）	FlatZinc是约束规划求解器的通用输入格式，用于描述变量、约束和目标函数。它基于MiniZinc高级建模语言编译生成，支持多种约束类型（如线性、逻辑），便于不同求解器（如Gecode、Chuffed）解析。
完备算法（Complete Algorithm）	完备算法（如分支定界、回溯搜索）能保证在有限时间内找到最优解或证明问题无解。其核心是系统性地遍历搜索空间，适用于需要严格解的场景（如数学证明、关键调度）。
非完备算法（Incomplete Algorithm）	非完备算法（如局部搜索、大邻域搜索）无法保证找到最优解或证明无解，但能在短时间内给出近似解。适用于大规模或实时性要求高的问题（如物流路径优化）。
不可满足核（Unsatisfiable Core）	目标函数的某个子集，该集合的理想情况取值在约束条件下是不可行的。例如，假设 C 为所有约束集合，目标函数为最小化布尔变量 a, b, c的和，若 $\bigwedge_{c \in C} c \wedge a = 0 \wedge b = 0$ 不可行，则可认为 $\{a, b\}$ 为一个不可满足核。
域（Domain）	变量的域是其所有可能取值的集合。例如，整数变量 $x \in \{1, 3, 5\}$ 的域包含三个值。求解器通过约束传播不断缩小域，直到域为单值（找到解）或空（无解）。
传播与推理（Propagation and Reasoning）	传播与推理是约束规划求解的核心技术，通过推理规则减少变量域。例如，若约束 $x = y + 1$ 且 $y \in \{1, 2\}$，则传播可推导出 $x \in \{2, 3\}$。
回溯	当搜索路径导致冲突时，回溯算法撤销部分赋值并尝试其他分支。例

（Backtrack）	如，在数独求解中，若某数字导致矛盾，则回溯到上一个选择点。
非时序回溯（Non-Chronological Backtrack）	一种改进的回溯策略，它不严格按照变量赋值的顺序回退，而是通过冲突分析直接跳转到导致矛盾的关键决策点。例如，在搜索过程中，若发现当前部分赋值与某约束冲突，求解器会识别冲突涉及的变量，并回溯到其中最新的一个（而非上一个赋值点）。
分支（Branching）	分支是搜索树中选择变量并赋值（或缩小变量的域）的过程。主要的决策点包括：变量选择（如优先选择容易导致冲突的变量、优先选域最小的变量）和赋值选择（如优先尝试最小值、尝试此前最深的搜索树中的赋值）。
解空间（Solution Space）	解空间是所有满足约束的变量赋值的集合。其大小随变量和域呈指数增长。求解器通过约束传播和搜索剪枝高效探索解空间，避免穷举。解空间的复杂性直接影响问题难度。
树搜索（Tree Search）	树搜索是约束规划中系统化遍历解空间的方法，通过分支（选择变量赋值）和回溯（撤销冲突赋值）构建搜索树。策略包括深度优先、广度优先或启发式引导（如最小剩余值）。树搜索的效率和完备性依赖于剪枝技术和变量/值选择策略，常用于精确求解组合优化问题。
对称性（Symmetry）	问题中存在的等价解（如排列对称性）。通过对称性消除（如固定变量顺序）可减少搜索空间，提升求解效率。
冲突分析（Conflict Analysis）	在搜索过程中识别导致冲突的约束，分析导致冲突的原因，并生成学习子句或约束以避免重复无效的搜索，避免陷入相同的冲突中。
子句学习（Clause Learning）	基于冲突分析生成有效子句或约束的过程，将学习子句加入到子句或约束库中可避免重复无效的搜索，是SAT求解器及其与CP结合的核心技术。
重启（Restart）	定期放弃当前搜索树重新开始，保留学习子句但重置变量赋值。结合随机化打破搜索偏置，是现代SAT、CP求解器中的常用技术。

B.3 黑箱优化术语

术语名称	**术语解释**
混合类型参数（Mixed-Type Parameters）	黑箱优化中同时包含连续型、离散型、类别型的参数组合，是工业实际优化场景的典型参数形式。
采集函数（Acquisition Function）	贝叶斯优化中依据概率模型评估候选点采样价值、平衡探索与利用、指引迭代采样的核心函数。
探索与利用（Exploration and Exploitation）	黑箱优化核心权衡逻辑，探索指拓展未知解空间，利用指深挖优质解区域，二者配比决定寻优效果。
超体积指标（Hypervolume Indicator）	量化多目标优化帕累托前沿优劣的核心指标，用于引导寻优方向、评估多目标优化的求解质量。
约束支配机制	带约束黑箱优化中，筛选可行解、剔除违规解的核心规则，保障优

（Constraint Domination Mechanism）	化结果满足预设约束条件。
维度灾难（Curse of Dimensionality）	高维优化场景下，参数空间随维度指数级膨胀，导致传统算法采样效率、寻优精度急剧下降的问题。
高斯过程（Gaussian Process，GP）	贝叶斯优化的核心概率代理模型，可预测未知函数值并量化不确定性，是拟合黑箱函数的关键工具。
拉丁超立方采样（Latin Hypercube Sampling，LHS）	黑箱优化的初始采样策略，以少量样本实现解空间的均匀覆盖，平衡初始评估成本与空间探索效率。
保真度（Fidelity）	在黑箱优化中，指代理模型、近似计算或简化仿真对真实目标函数的还原精准程度；保真度越高，模型预测与真实函数评估的偏差越小，直接决定寻优结果的可靠性。
代理模型（Surrogate Model）	针对计算昂贵、无显式解析表达式、梯度不可得的目标/约束函数，通过少量高精度仿真或实验样本训练得到的快速近似数学模型，用于在优化迭代中替代原高耗时模型进行函数评估，从而大幅降低优化计算成本、加速寻优收敛。

B.4 数值计算术语

术语名称	**术语解释**
浮点数与机器精度（Floating Point & Machine Precision）	浮点数是计算机中表示实数的标准格式，通常遵循 IEEE 754 标准。机器精度（Machine Epsilon）是计算机可区分的最小相对数值差，反映了计算系统的极限分辨能力，是误差分析的重要基础。
舍入误差与截断误差（Round-off & Truncation Error）	舍入误差来源于浮点表示的有限精度；截断误差来源于无限数学过程（如级数、积分或导数）被有限步骤近似。数值计算的总误差通常由这两类误差叠加形成。
条件数（Condition Number）	衡量数学问题对输入扰动敏感程度的指标。条件数大的问题称为病态问题（Ill-conditioned Problem），小的输入误差可能导致输出结果的巨大变化。条件数反映问题本身的难度，而非算法性能。
数值稳定性（Numerical Stability）	描述算法在计算过程中对误差放大的抑制能力。数值稳定算法不会显著放大舍入误差，即使在有限精度环境中也能保持结果可靠。
收敛性与收敛阶（Convergence & Order of Convergence）	若数值算法产生的近似解序列逐步逼近真实解，则称算法收敛。收敛阶描述误差减少速度，如线性收敛、二次收敛、超线性收敛等，是评价迭代算法效率的重要指标。
预处理（Preconditioning）	通过变换原问题以改善其数值性质，从而加速迭代算法收敛的技术。预条件矩阵可以显著降低条件数，是大规模线性系统求解的关键策略。
残差与残差范数	残差是近似解代入原方程后产生的误差向量；残差范数是其大小

（Residual & Residual Norm）	度量（如2范数或无穷范数）。残差范数常作为迭代算法的收敛判据，但小残差不一定意味着小真实误差。
绝对误差与相对误差（Absolute & Relative Error）	绝对误差是数值近似值与真实值之差的绝对值；相对误差是绝对误差与真实值绝对值的比值。绝对误差反映偏差大小，相对误差反映误差在尺度上的相对影响，是数值算法精度评价的常用指标。
良态问题与病态问题（Well-Conditioned & Ill-Conditioned Problem）	若输入数据的微小扰动仅导致输出结果的微小变化，则问题为良态问题；若微小扰动会被显著放大，则为病态问题。病态性由问题本身决定，通常通过条件数衡量，而非算法设计造成。
适定问题与不适定问题（Well-Posed & Ill-Posed Problem）	适定问题满足——解存在、解唯一、解对输入连续依赖。不满足任一条件的问题称为不适定问题。不适定问题常见于反问题与信号重建，需要正则化或约束恢复稳定解。
稀疏矩阵与稠密矩阵（Sparse & Dense Matrix）	稀疏矩阵中大多数元素为零，可采用压缩存储结构（CSR/CSC 等）减少存储和计算开销；稠密矩阵大部分元素非零，通常采用连续存储。稀疏性是大规模科学计算的核心结构特征。
数值离散（Numerical Discretization）	将连续数学模型（微分方程、积分方程等）转化为有限维离散问题的过程，例如有限差分、有限元和有限体积方法。离散精度与稳定性直接影响数值解质量。
矩阵分解（Matrix Factorization）	将矩阵表示为多个结构更简单矩阵的乘积，如 LU、QR、Cholesky、SVD 分解。矩阵分解是线性系统求解、最小二乘拟合和特征值计算的基础数值工具。
矩阵谱（Matrix Spectrum）	矩阵谱是矩阵所有特征值的集合。谱分布决定了线性系统的稳定性、迭代收敛速度及微分方程离散系统的动力学性质，是谱分析和稳定性理论的核心概念。
负载均衡（Load Balancing）	在并行计算中合理分配任务与数据，使各计算节点工作量均衡，减少空闲等待时间。良好的负载均衡是高并行效率和可扩展性的关键。
可扩展性（Scalability）	算法或系统在增加计算资源（CPU/GPU/节点数）时性能提升的能力。可扩展性衡量并行效率，是评估大规模数值计算软件的重要指标。
稳定性（Stability）	算法在计算过程中对舍入误差或输入数据扰动的敏感程度。如果初始误差在迭代过程中不会无限放大，则称算法是稳定的。这是数值软件能否在真实计算机上运行的前提。
相容性（Consistency）	描述离散算子与连续算子之间的接近程度。当步长（如时间步 h 或网格间距 Δx）趋于零时，离散方程若能还原为原始的微分方程，则称该格式具有相容性。
刚性（Stiffness）	描述方程中存在多个时间尺度，且尺度差异极大（例如化学反应中某些组分反应极快，某些极慢）。求解刚性方程通常需要使用显式方法之外的隐式方法，否则必须使用极小的步长。
CFL条件（Courant-Friedrichs-	偏微分方程数值求解中的稳定性必要条件。它要求数值依赖域必须包含物理依赖域，通俗来说就是：数值信息的传递速度必须大

Lewy Condition）	于或等于物理波的传播速度。
雅可比矩阵（Jacobian Matrix）	向量值函数的一阶导数矩阵。在多元非线性系统中，它决定了搜索的方向。
海森矩阵（Hessian Matrix）	标量函数（如能量函数）的二阶导数矩阵。用于描述函数的曲率，是二阶优化算法（如牛顿法）的核心。
正交变换（Orthogonal Transformation）	如 Householder 变换和 Givens 旋转。它们在不改变向量模长的情况下进行消元，是数值稳定性最高的矩阵运算手段。
先验误差（A Priori Error）	先验误差是指在计算实际开始之前，基于算法本身的性质、问题的数学特征以及离散化参数（如步长或矩阵性质），对数值解与精确解之间差异的理论估计。
后验误差（A Posteriori Error）	后验误差是指在完成数值计算之后，利用已经得到的数值解（近似解）以及已知的数据（如方程的右端项、系数等）来推导出的误差估计。

B.5 AI辅助求解术语

术语名称	**术语解释**
冷启动（Cold Start）	在没有任何可利用的历史求解数据或先验经验的条件下，为一个新问题或新环境寻找有效配置算法。
图神经网络（Graph Neural Network）	一种专门处理图结构数据的深度学习架构，在优化中常将变量与约束的关系表示为二分图进行处理。
模仿学习（Imitation Learning）	一种机器学习技术，其核心思想是通过观察并模仿“专家”的行为来高效训练智能体。
强化学习（Reinforcement Learning）	一种机器学习技术，其核心思想是通过智能体与环境（如求解器状态）的交互，根据获得的奖励（如Gap值缩小、时间缩短）不断优化决策策略的过程。
强分支（Strong Branching）	一种高质量但计算昂贵的分支决策启发式。它通过对候选变量试探性地进行局部求解，预先观察分支后的改进程度来选出最佳变量。
进化算子（Evolutionary Operator）	在演化算法中用于产生新个体的操作，常见的进化算子包括变异（对单个个体进行局部修改）和交叉（融合两个优秀个体片段）。
提示工程（Prompt Engineering）	通过设计、优化和精炼输入给大语言模型的提示词，让大模型更好地理解用户意图，引导其生成更准确、相关且高质量的输出。
适应度（Fitness）	演化算法中衡量算法个体优劣的量化指标。个体适应度越高代表其解决问题的质量越高，将有更大概率被算法选中进入“下一代”。
最优解保留（Optimal Solution	一种将当前发现的最好的一个（或几个）解，不经过任何修改直接复制到下一代中，确保算法性能“不退化”的策略

Preservation，OSP）	
帕累托最优（Pareto Optimality）	在多目标优化中的一种极限优化状态，即无法在不损害至少一个其他目标的前提下，让任何一个目标变得更好。
帕累托前沿（Pareto Front）	在多目标优化中，无法在不损害一个目标的前提下改进另一个目标的所有帕累托最优解集合，代表了最优的权衡方案集。

B.6 领域求解术语

术语名称	术语解释
领域求解器（Domain-Specific Solver）	领域求解器是面向特定行业复杂决策难题的专用优化引擎。它将通用数学规划与优化算法能力，同具体业务场景的深层逻辑与领域约束相结合，从而高效求解供应链多环节和领域中的NP难问题。
启发式算法（Heuristic Algorithm）	一种基于经验、直觉或近似规则的优化算法，它在可接受的时间和空间范围内，快速寻找待解决问题的一个满意解（近似最优解），在可接受时间内给出 NP-hard 组合优化问题的可行满意解，是求解大规模复杂优化问题的基础方法，核心优势为计算效率高、工程落地性强。
元启发式算法（Metaheuristic Algorithm）	元启发式算法是一种高层次通用型优化框架，不依赖问题特定结构，通过迭代搜索策略跳出局部最优，覆盖全局解空间，是求解 NP-hard 问题的主流算法体系。
邻域搜索（Neighborhood Search）	邻域搜索是基于局部改进的迭代优化算法，从初始可行解出发，通过定义解空间邻域结构，在邻域内寻找更优解并迭代更新，是绝大多数现代启发式算法的核心底层逻辑。
邻域算子（Neighborhood Operator）	邻域算子是定义邻域结构的基本操作单元，通过交换、插入、删除、反转、位移等操作生成当前解的邻域解，直接决定邻域搜索的搜索方向与效率。
变邻域搜索（Variable Neighborhood Search）	变邻域搜索通过系统性切换不同邻域算子与搜索结构，在全局搜索与局部开发间动态平衡，避免算法早熟收敛，是求解组合优化问题的高性能元启发式算法。
遗传算法（Genetic Algorithm）	遗传算法是模拟生物进化的进化类元启发式算法，通过选择、交叉、变异操作迭代优化种群，具备全局搜索能力，在解空间搜索近似最优解。
模拟退火算法（Simulated Annealing Algorithm）	模拟退火算法是以一定概率接受劣解跳出局部最优，全局搜索能力强，对初始解依赖性低，是经典的单解迭代型元启发式算法。
禁忌搜索算法（Tabu Search Algorithm）	禁忌搜索是基于记忆机制的智能局部搜索算法，通过禁忌表记录近期搜索行为避免循环回溯，引入渴望准则突破最优解，是求解调度类问题的经典高性能算法。
LKH算法	一种高性能的TSP启发式算法，通过动态自适应k阶邻域变换实

（Lin-Kernighan-Helsgaun Heuristic Algorithm）	现高精度搜索，是目前求解TSP问题的最优启发式算法之一。
自适应大邻域搜索算法（Adaptive Large Neighborhood Search Algorithm）	在大邻域搜索基础上引入自适应算子选择机制，根据历史性能动态加权优选破坏/修复算子，全局搜索能力强、鲁棒性高。
三维装箱问题（3D Bin Packing Problem）	给定一系列具有长、宽、高三维尺寸的长方体物品和容量固定的标准箱体，在满足物品不可分割、不可重叠、必须完全放入箱体内等约束下，寻找使用箱体数量最少的装箱方案。
二维排样问题（2D-Nesting Problem）	二维排样问题，指在给定的板材或区域内，以最大化材料利用率目标，将多个零件进行无重叠、合理排列的组合优化问题。在服装裁剪、钣金加工、家具制造制造行业有广泛应用。
临界多边形（No-Fit Polygon，NFP）	二维不规则零件排样中用于表示两个简单多边形相对位置关系的几何工具。它定义为当一个多边形围绕另一个固定多边形无滑动旋转一周时，其参考点所形成的轨迹，用于快速判断两个多边形是否重叠以及寻找最优贴合位置。
重叠移除（Overlap Removal）	二维不规则排样算法中的关键步骤，通常在初始布局生成后进行局部调整，以消除零件间的重叠并优化布局质量。
作业车间调度问题（Job-Shop Scheduling Problem，JSP）	生产调度中的另一个经典问题，每个作业的加工顺序可以不同，每个作业由多个操作组成，每个操作需要在特定的机器上完成，且每个作业的操作顺序是固定的，目标通常是最小化最大完成时间。
柔性作业车间调度问题（Flexible Job-Shop Scheduling Problem，FJSP）	作业车间调度问题（JSP）的扩展，增加了机器的灵活性，每个作业的操作可以在多个机器中选择一个进行加工，而不是固定在某一台机器上。目标通常是最小化最大完成时间或其他性能指标。
流水车间调度（Flow Shop Scheduling，FSP）	所有工件按相同顺序经过一系列机器。
混合流水车间调度（Hybrid Flow Shop Scheduling，HFSP）	结合流水车间与并行机特点，某些阶段存在多台并行机器。
完工时间（Makespan）	最后一个工序完成的时间。
旅行商问题（Travelling Salesman Problem）	给定一组城市和每对城市之间的距离，求解单一车辆访问每座城市一次并回到起始城市的最短回路。
车辆路径问题（Vehicle Routing Problem）	给定一组车辆和客户，求解最优的配送路线组合。
带容量限制的车辆路径问题（Capacitated Vehicle	在车辆路径问题基础上，给定每个车辆的容量，即最大载重，求解满足容量约束下的最优路线组合。

Routing Problem）	
带时间窗的车辆路径问题（Vehicle Routing Problem with Time Windows）	在车辆路径问题基础上，给定每个客户的可访问时间段，求解满足客户时间要求下的最优路线组合。
多仓配送的车辆路径问题（Multi-Depot Vehicle Routing Problem）	在车辆路径问题基础上，额外给定多个配送中心（Depot），求解最优仓库与路线组合。
取送货问题（Pickup and Delivery Problem）	在车辆路径问题基础上，给定若干成对“取货-送货”客户，在满足车辆容量约束及访问先后约束下，求解最优配送路线组合。
高级生产计划与排程（Advanced Planning and Scheduling，APS）	一种基于约束理论、运筹学和人工智能技术的先进生产管理系统，通过集成生产计划、资源调度和供应链协同，在满足多重约束（设备能力、人力、物料、交货期）的前提下，自动生成最优生产方案，实现准时交付、最大化产能利用率和最小化库存成本的目标，是连接ERP与MES系统的核心枢纽。
需求预测（Demand Forecasting，DF）	需求预测是供应链与运筹优化的前端核心模块，通过时间序列分析、因果推断、统计拟合等方法，依托机器学习与深度学习模型，融合外部特征（节假日、天气、政策、舆情），实现多周期、多品类预测，是数字供应链与智能计划系统的核心组件。
库存计划（Inventory Planning，IP）	库存计划是在需求预测基础上，以总成本最小（订货成本 + 持有成本 + 缺货成本）为目标，确定最优订货批量、订货点、安全库存与补货周期的运筹优化问题。
主生产计划（Master Production Schedule，MPS）	主生产计划是连接综合生产规划与车间作业排程的核心枢纽，基于需求预测与库存水平，确定最终产品的生产数量、生产时段、交付节点，平衡订单交付、设备产能和库存水平。
物料需求计划（Material Requirement Planning，MRP）	物料需求计划是一种基于主生产计划、产品物料清单和库存信息，通过逆向推算来确保物料按时、按量供应的核心管理方法，精确计算出何时需要采购或生产多少原材料、零部件，是制造业资源调度的基础逻辑。
有限能力批量计划问题（Capacitated Lot Size Problem，CLSP）	CLSP 是生产计划领域经典的 NP-hard 优化问题，核心是在多周期、多产品环境下，考虑生产能力约束、库存持有成本、生产准备成本和缺货成本，确定各周期的最优生产批量，以最小化总运营成本。
多级有限能力批量计划问题（Multi-Level Capacitated Lot Sizing Problem）	指CLSP的扩展，考虑了产品具有多级物料清单结构（BOM）的情形。它不仅需要决定最终产品和各级零部件的生产批量与时机，还需协调上下级物料之间的供需依赖关系。
物料清单（Bill of Materials，BOM）	描述产品结构的核心数据文件，以层级化方式详细列出构成最终产品的所有原材料、半成品、零部件、附件及其数量关系和装配顺序，是连接产品设计、生产计划、采购、库存管理和成

	本核算的产品图谱，为MRP和APS系统提供基础数据支撑。
拉格朗日松弛（Lagrangian Relaxation）	一种用于求解复杂约束优化问题的分解方法。其核心是将模型中造成问题困难的约束“松弛”到目标函数中，并引入拉格朗日乘子对其进行惩罚，从而将原问题分解为若干个更易求解的子问题。
列生成（Column Generation）	一种用于求解包含极多变量（列）的大规模线性规划问题的高效算法。通过求解一个或多个定价子问题来动态生成有潜力的新变量（列）加入主问题，从而渐进地逼近最优解。
松弛-固定算法（Relax-and-Fix Algorithm）	一类用于获取大规模MILP问题的可行解的启发式框架，通过先松弛部分整数变量为连续变量求解，然后依次固定一部分关键整数变量的值，逐步缩小问题规模。
固定-优化算法（Fix-and-Optimize Algorithm）	一种专门用于求解大规模MILP的启发式算法框架，在获得一个初始可行解的情况下，在每次迭代中固定部分整数变量的值，松弛一个变量子集重新进行优化求解优化。
路径重连算法（Path-Relinking Algorithm）	一种增强的邻域搜索策略，通过两个或多个可行解之间考察解相互之间的关联路径，以发现结合不同引导解优良特性的新解，从而更深入地挖掘有效邻域，提高算法的搜索效率和解的质量。

华为技术有限公司
深圳龙岗区坂田华为基地
电话：+86 755 28780808
邮编：518129
www.huawei.com

商标声明

免责声明

本文档可能含有预测信息，包括但不限于有关未来的财务、运营、产品系列、新技术等信息。由于实践中存在很多不确定因素，可能导致实际结果与预测信息有很大的差别。因此，本文档信息仅供参考，不构成任何要约或承诺，华为不对您在本文档基础上做出的任何行为承担责任。华为可能不经通知修改上述信息，恕不另行通知。